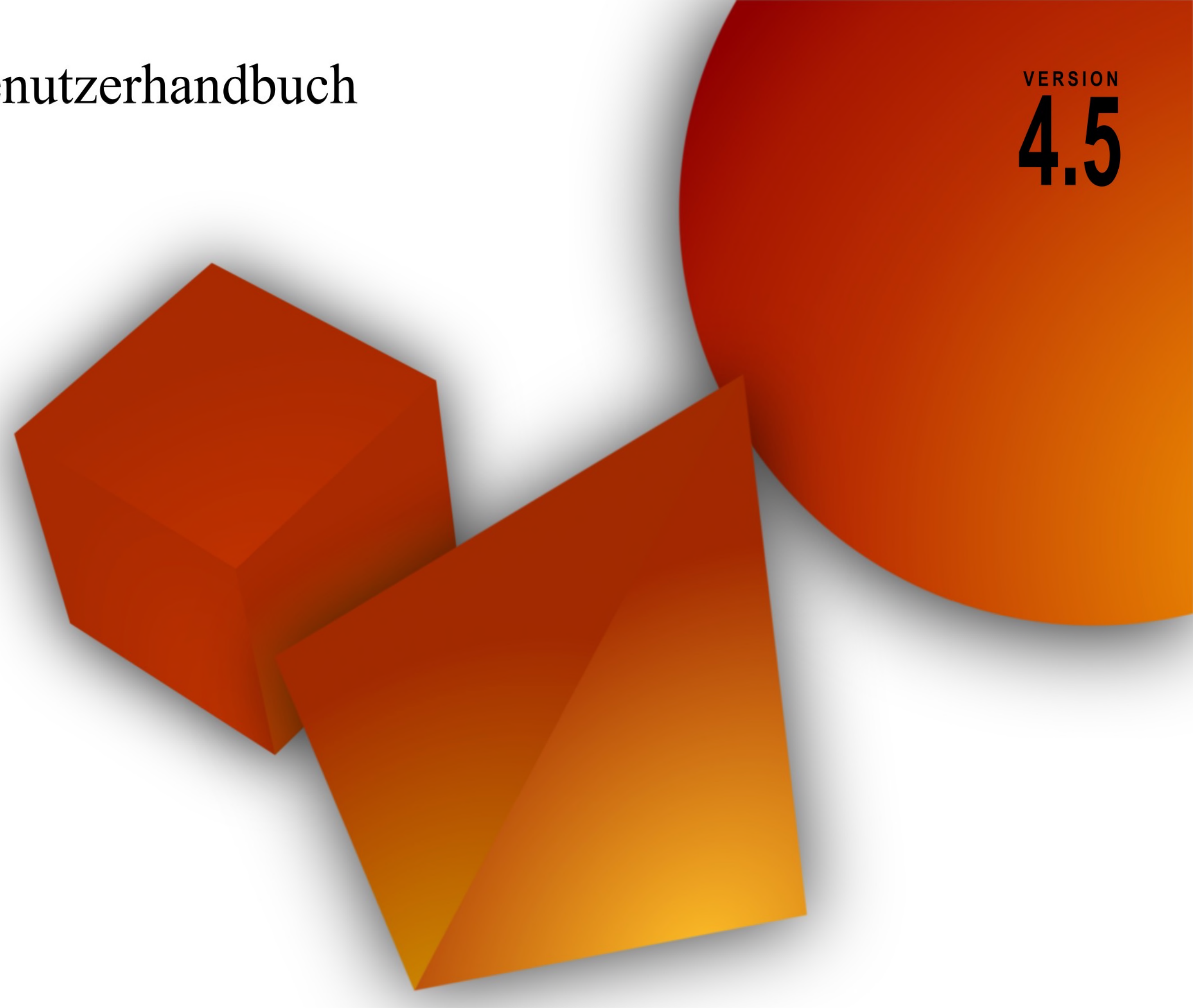
Benutzerhandbuch

VERSION 4.5



Borland® C++

Borland C++ 4.5

Benutzerhandbuch

BORLAND

Copyright © 1987, 1994 by Borland International, Inc.
Alle Rechte vorbehalten.

Borland hat möglicherweise Patente oder Patentanmeldungen für Teile des hier behandelten Themas. Dieses Handbuch gibt Ihnen keine Rechte auf diese Patente.

Alle Borland-Produkte sind Warenzeichen oder eingetragene Warenzeichen von Borland International, Inc. Andere aufgeführte Waren- und Produktbezeichnungen sind Warenzeichen oder eingetragene Warenzeichen der jeweiligen Rechteinhaber.

Einschränkung der Gewährleistung:
Es wird keine Garantie für die Richtigkeit des Inhaltes dieses Handbuches übernommen. Da sich Fehler, trotz aller Bemühungen, nie vollständig vermeiden lassen, sind wir für Hinweise jederzeit dankbar.

Herausgeber:

Borland GmbH
Monzastraße 4c
63225 Langen

Satz, Layout & Acrobat-Bearbeitung:

ALFES Computers, Publishing & Graphics, Siegen

Printed in Ireland.

Inhalt

Einführung . 17
Was ist neu in Borland C++? 17
Neu in Version 4.0 18
Hardware- und Software-Anforderungen 19
Konventionen . 19
Hilfesystem . 20
Durchsuchen mehrerer Hilfedateien mit OpenHelp . 21
Wie erreichen Sie Borland? 21

Teil I
Verwenden der integrierten Entwicklungsumgebung

Kapitel 1
Erste Schritte . 27
Borland C++ installieren 27
Hardware- und Softwareanforderungen 28
Installationsschritte 28
Borland C++ starten 30
Hilfe aufrufen . 32
Die IDE konfigurieren 32
Mauspaletten ändern 33
IDE-Vorgaben setzen 34
IDE-Vorgaben speichern 35
Einsatz des Editors 35
IDE-Editor konfigurieren 36
Syntaxhervorhebung 36
Meldungsfenster . 37
Programm durchsehen 38
Objekte durchsehen (Klassenübersicht) 39
Deklarationen aufgelisteter Symbole anzeigen 40
Globale Symbole anzeigen 40
Mit regulären Ausdrücken in der Symbolanzeige suchen 42
Symbole im eigenen Programm anzeigen . 42

Kapitel 2
Erstellen und Verwalten von Projekten . 43
Was ist Projektverwaltung? 43
Verwendung des Projekt-Managers 44
Anlegen eines Projekts 46
Setzen von Optionen für Zieldateien mit dem Dialogfenster *Neues Ziel* . 46
Festlegung des Typs bestimmender Knoten 50
Öffnen bestehender Projekte 50
Hinzufügen von bestimmenden Knoten zu einem Objekt 50
Löschen von bestimmenden Knoten . 51
Bearbeiten der Eigenschaften bestimmender Knoten 51
Hinzufügen von Zielknoten zu einem Projekt . 52
Zielknoten löschen 53
Bearbeiten von Zieleigenschaften mit TargetExpert . 53
Verschieben von Knoten in einem Projekt . 53
Knoten kopieren 54
Umwandeln von Projektdateien in Make-Dateien . 55
Importieren von Projekten aus Borland C++ 3.x 55
Anpassen des Projektfensters 55
Gruppieren von Dateien mit Quelltextsammlungen 57
Erstellen einer Quelltextsammlung 58
Übersetzer, Viewer und Hilfsprogramme . . 58

Hinzufügen von Übersetzern, Viewern und Hilfsprogrammen zur IDE 60

Kapitel 3
Bestimmen von Projektoptionen und Compilierung 63

Einstellen der Projektoptionen 63
Benutzen von Schablonen 64
Vordefinierte Schablonen 65
Standard-Projektoptionen 65
Schablonen verwalten 66
Verbinden einer Schablone mit einem Knoten 67
Mehrfache Benutzung von Schablonen 67
Dateien der Projektbeschreibungssprache 68
Setzen von überschreibenden lokalen Optionen 69
Ansehen der Optionen eines Projekts 71
Projekte compilieren 72
Compilieren von Teilen eines Projekts 73
Beheben von Compilierzeit-Fehlern 74
Fehler ansehen 74
Fehlerbehebung 75
Compilieren in der IDE 75
IDE Compiler Optionen 75
Kommandozeilen-Compiler 76
Konfigurationsdateien 77
Response-Dateien 77
Prioritätsregeln bei Optionen 78
Compiler-Optionen-Tabelle 78
Verzeichnisse 86
Datei-Such Algorithmus 87
Compiler definiert 88
Compiler/Code-Erzeugung 88
Compiler/Gleitkomma 90
Compiler/Compilerausgabe 90
Compiler/Quelltext 90
Compiler/Debugger 91
Compiler/Vorcompilierte Header 92
16-Bit Compiler/Prozessor 93
16-Bit Compiler/Aufrufkonvention 93
16-Bit Compiler/Speichermodell 94
16-Bit Compiler/Datensegmentnamen für Segmente 96
16-Bit Compiler/FAR-Datensegmentnamen 97
16-Bit Compiler/Namen für Code-Segmentnamen 97
16-Bit Compiler/Entry/Exit-Code 97
32-Bit Compiler/Prozessor 99
32-Bit Compiler/Aufrufkonvention 100
C++ Optionen/Elementzeiger 100
C++ Optionen/C++ Kompatibilität 101
C++-Optionen/Virtuelle Tabellen 102
C++-Optionen/Templates 103
C++- Optionen/Exceptions/RTTI 103
Optimierungen 104
Optimierungen/Spezielle Optimierungen . . 104
Optimierungen/Größe 105
Optimierungen/Geschwindigkeit 106
Warnungen 109
Warnungen/Portabilität 110
Warnungen/ANSI Violations 110
Warnungen/Obsolete C++ 110
Warnungen/Potentielle C++-Fehler 111
Warnungen/Ineffiziente C++-Codierung . . 111
Warnungen/Potentielle Fehler 111
Warnungen/Ineffiziente Codierung 112
Warnungen/Allgemein 112
Make 112
Kommandozeilenoptionen 112

Kapitel 4
Anwendungen anlegen mit AppExpert 117

Grundlagen zu AppExpert 118
Erstellen einer kleinen Anwendung 119
Übersicht über den erzeugten Code . . . 121
Das Anwendungsobjekt 121
Client-Fenster 125
Überblick über die Optionen von AppExpert 125
Anwendung 125
Grundoptionen 126

Optionen für OLE 2 127
Code-Generierung 128
Verwaltungsoptionen 129
HauptfensterOptionen 129
Grundoptionen 129
SDI Client . 130
MDI Client 132
Dialog Client 132
MDI Child/View-Optionen 133
Grundoptionen 133

Kapitel 5
Anwendungen verändern mit ClassExpert . 135

ClassExpert starten 135
Grundlagen zu ClassExpert 136
Klassenausschnitt 136
Ereignisausschnitt 136
Bearbeitungsausschnitt 136
Hinzufügen einer Klasse 137
Erstellen von Dokumenttypen 138
Hinzufügen und Löschen von Ereignisbehandlungsroutinen (Event-Handlern) 140
Hinzufügen und Löschen von Instanzvariablen 141
Zum Quelltext einer Klasse wechseln . . 142
Die Verwendung von Resource Workshop unter ClassExpert 143
Starten von Resource Workshop aus der IDE . 143
Verwenden von Rescan 144
Löschen einer Klasse 144
Verschieben einer Klasse 145
Umbenennen eines Elements 145
Importieren einer Klasse 145
Wiederanlegen einer .APX-Datenbankdatei 146

Kapitel 6
Debuggen in der IDE 147

Arten von Fehlern 147
Compilierzeitfehler 147
Laufzeitfehler . 149
Logische Fehler 149
Erzeugen von Debug-Informationen 149
Kommandozeilenparameter beim Debuggen . 150
Steuerung des Programmablaufs 150
Beobachten der Programmausgaben . . . 150
Programm mit *Funktion überspringen* ausführen lassen 150
Programm mit *Nächste Anweisung* ablaufen lassen 152
Schrittweise Abarbeitung bei Elementfunktionen von Klassen . . . 152
Schrittweise Abarbeitung bei externen Programmteilen 152
Große Programmteile mit *Funktion überspringen* überspringen 153
Lauf bis zu einem bestimmten Punkt . 153
Funktion suchen 153
Rückkehr zur aktuellen Ausführungsposition 153
Rückwärts gehen 154
Anhalten des Programms 154
Neu starten . 155
Werte abfragen . 155
Was ist ein Ausdruck? 155
Ausdrücke überwachen 155
Formatieren von überwachten Ausdrücken . 156
Deaktivieren eines überwachten Ausdrucks . 158
Löschen eines überwachten Ausdrucks . 158
Editieren eines überwachten Ausdrucks . 158
Berechnen und Verändern von Ausdrücken . 158
Auswerten von Ausdrücken 158
Modifizieren von Variablen 160
Datenelemente ansehen 160
Registerwerte inspizieren 162
Haltepunkte verwenden 162
Haltepunkte setzen 162
Arbeiten mit Haltepunkten 163
Haltepunkte löschen 163

Aktivieren und Deaktivieren von Haltepunkten 164
Ansehen und Editieren des Programms an einer Haltepunkt-Position 164
Ungültige Haltepunkte löschen 164
Ansehen und Ändern von Haltepunkt-Attributen 165
Setzen eines neuen Haltepunkts . 165
Modifizieren eines vorhandenen Haltepunkts 165
Setzen von Haltepunkten, während das Programm läuft . . . 165
Bedingte Haltepunkte setzen . . . 166
Anpassung der Farben für Haltepunkte und Ausführungszeiger . . . 166
Abfangen des Fehlers Allgemeine Schutzverletzung . 167
Benutzen des Fensters Ereignisse 167
Debuggen von dynamischen Linkbibliotheken (DLLs) 168
Soft- und Hard-Modus des Debuggers 169

Kapitel 7

WinSight . 171
WinSight starten . 171
Bildschirmaktualisierung freigeben und anhalten . 172
Botschaftsverfolgung abschalten . . . 172
Einen Ausschnitt auswählen 172
Klassenliste . 173
Den Ausschnitt *Klassenliste* benutzen . 173
Klassen untersuchen 174
Fensterhierarchie . 174
Ein Fenster finden 175
Den Modus *Fenster auswählen* verlassen . 175
Fenster überwachen 176
Zu überwachende Botschaften filtern 176
Mit dem Ausschnitt *Botschaften* arbeiten . 176

Teil II
Verwendung der Kommandozeilen-Hilfsprogramme

Kapitel 8

WinSpector . 187
Mit WinSpector arbeiten 187
WINSPCTR.LOG konfigurieren 188
WINSPCTR.LOG Referenz 189
Abschnitt Disassemblierung 191
Abschnitt Stack-Trace 191
Abschnitt Register 192
Abschnitt Botschaftswarteschlange 192
Abschnitt Tasks 192
Abschnitt Module 193
Abschnitt USER- und GDI-Heap-Information 193
Abschnitt Systeminformation 194
Verarbeiten der Daten von WinSpector . . . 194
DFA-Ausgabe 195
DFA und WINSPCTR.LOG 195
DFA und WINSPCTR.BIN 195
Weitere Dienstprogramme von WinSpector . 196
Arbeiten mit EXEMAP.EXE 196
Arbeiten mit TMAPSYM.EXE 197
Arbeiten mit BUILDSYM.EXE 197

Kapitel 9

Arbeiten mit dem Linker: TLINK 199
Grundsätzliches zu TLINK 199
TLINK.CFG . 201
Response-Dateien 201
TLINK mit BCC.EXE benutzen 202
Bibliotheken linken 202
Optionen für TLINK 204
Die Moduldefinitionsdatei 212
Anweisung CODE 213
Anweisung DATA 213
Anweisung DESCRIPTION 214
Anweisung EXETYPE 214
Anweisung EXPORTS 214
Anweisung IMPORTS 215

Anweisung LIBRARY 216
Anweisung NAME 217
Anweisung SEGMENTS 217
Anweisung STACKSIZE 218
Anweisung STUB 218
Voreinstellungen für
Moduldefinitionsdateien 219
Anweisung HEAPSIZE 219

Kapitel 10
Arbeiten mit den
Ressourcen-Dienstprogrammen 221

BRCC.EXE: Der Ressourcencompiler 222
RLINK: Der Ressourcen-Linker 223
BRC.EXE: Die Bedienungsoberfläche
für die Ressourcen-Dienstprogramme 224

Kapitel 11
Bibliotheken . 227

IMPLIB: Verwaltung von
Importbibliotheken 227
IMPDEF: Verwaltung von
Moduldefinitionsdateien 228
Klassen in einer DLL 229
Funktionen in einer DLL 230
TLIB: Bibliotheksverwaltung 231
Gründe für den Einsatz von
Objektmodul-Bibliotheken 231
Die TLIB-Kommandozeile 231
Response-Dateien 232
Groß-/Kleinschreibung:
Die Option
/CTLIB;Groß-/Kleinschreibung 233
Ein erweitertes Verzeichnis
erstellen:
Die Option /E 233
Die Seitengröße definieren:
Die Option /PTLIB; Seitengröße . . . 233
Kommentare entfernen:
Die Option /0TLIB;Kommentare
entfernen . 234
Die Liste der Operationen 234
Beispiele . 235

Kapitel 12
Mit MAKE arbeiten 237

Arbeitsweise von MAKE 237
BUILTINS.MAK 239
Arbeiten mit TOUCH.EXE 239
Optionen für MAKE 239
Optionen zu
Standardvoreinstellungen machen . . 241
Kompatibilität zu NMAKE von
Microsoft . 241
Arbeiten mit Make-Dateien 242
Symbolische Zielangaben 242
Regeln für symbolische Ziele 243
Explizite und implizite Regeln 243
Syntax für explizite Regeln 243
Mehrere Regeln für ein Ziel 244
Syntax für implizite Regeln 245
Explizite Regeln mit impliziten
Befehlen . 246
Befehlssyntax . 246
Befehls-Präfixe 246
Verwendung von @ 247
Verwendung von num und - 247
Verwenden von & 247
Befehlsoperatoren 247
Fehlersuche mit Hilfe von
temporären Dateien 248
Verwenden von MAKE-Makros 248
Makros definieren 249
Makros aufrufen 249
Ersetzen von Strings in Makros 250
Vordefinierte MAKE-Makros 250
Verändern vordefinierter Makros 251
Verwenden von MAKE-Direktiven 251
.autodepend . 252
!error . 252
Alle Optionen zum Abschalten
der Fehlerprüfung 253
!if und andere Bedingungsdirektiven . . 253
!include . 254
!message . 255
.path.ext . 255
.precious . 255
.suffixes . 256

!undef ... 256
Makros innerhalb von Direktiven verwenden ... 257
Leere Makros ... 257

Kapitel 13
Die Grundlagen des Resource Workshop ... 259
Was Windows-Ressourcen sind ... 259
Ressourcen-Dateitypen ... 262
Bitmap-Ressourcen-Dateien ... 263

Teil III
Verwenden des Resource Workshop

Kapitel 14
Arbeiten mit Projekten, Ressourcen und Bezeichnern ... 267
Ein neues Projekt erzeugen ... 267
Öffnen eines vorhandenen Projekts ... 268
Die Verwendung des Projektfensters ... 269
Eingebundene und referenzierte Ressourcen ... 269
Informationen im Projektfenster anzeigen ... 269
Bezeichner anzeigen ... 270
Ressourcen anzeigen ... 270
Einträge anzeigen ... 270
Nicht verwendete Typen anzeigen ... 270
Auswählen einer Ressource ... 270
Arbeiten mit Ressourcen ... 271
Ressourcen öffnen ... 271
Ressourceneditoren ... 271
Der interne Texteditor ... 271
Ressourcen einfügen ... 272
Direktes Einbinden einer Ressource ... 272
Einfügen einer referenzierten Ressource ... 273
Verschieben einer Ressource ... 274
Ressourcen zwischen Projekten kopieren ... 274
Ressourcen löschen ... 275
Ressourcen umbenennen ... 275
Angeben von Speicheroptionen für Ressourcen ... 276
Bezeichner verwenden ... 276
Bezeichnerkomponenten ... 277
Bezeichnerdateien ... 278
Bezeichnerdateien erzeugen ... 278
C Header-Dateien ... 278
Automatische Bezeichnerverwaltung ... 279
Ohne Bezeichnerdatei arbeiten ... 280
Bezeichner zufügen ... 280
Umbenennen der Ressource. ... 280
Verwenden des Bezeichner-Dialogfensters ... 281
Bezeichner bearbeiten ... 281
Bezeichner löschen ... 281
Bezeichner auflisten ... 282
Starten eines Ressourceneditors ... 282
Vorgaben einstellen ... 283
Stufenweises Rückgängigmachen ... 283
Texteditor ... 283
Mehrfachspeichern ... 283
Windows Version ... 284
Arbeiten mit Binärdateien ... 284
32-Bit-Ressourcen erzeugen ... 285

Kapitel 15
Dialogfenster erzeugen ... 287
Der Aufruf des Dialogeditors ... 287
Eine neues Dialogfenster erzeugen ... 287
Bearbeiten eines bereits existierenden Dialogfensters ... 288
Benutzung des Dialogeditors ... 289
Ein Dialogfenster auswählen ... 289
Einstellen der Dialogfenster-Attribute ... 289
Titel hinzufügen ... 290
Ändern der Rahmenart ... 290
Eigenschaften ändern ... 290
Schriften ändern ... 291
Ein Menü einfügen ... 292
Zuweisen einer selbstdefinierten Klasse an ein Dialogfenster. ... 292
Festlegung der Dialogfensterposition ... 292
Arbeiten mit Dialogelementen ... 293

Dialogelement-gruppen 293
Die Werkzeugpalette 294
Dialogelemente markieren 296
Dialogelemente einem Dialogfenster hinzufügen 297
Mehrere Kopien eines Dialogelements in ein Dialogfenster einfügen 298
Eigenschaften der Dialogelemente (.VBX Elemente) 298
Dialogelemente verschieben 299
Die Größe von Dialogelementen verändern 299
Das gleichzeitige Positionieren und Verändern der Größe von Dialogelementen 299
Das Ausrichten von Dialogelementen mit Hilfe eines Gitters 300
Dialogelemente bearbeiten 300
Hinzufügen von Titeln zu Dialogelementen 301
Die Klasse eines Elements verändern 302
Bestimmen von Dialogelementen als Tabulatorpositionen 302
Das Gruppieren zusammenhängender Dialogelemente 303
Dialogelemente umordnen 303
Ausrichten, Anordnen und Größe verändern von Dialogelementen 304
Mehrere Dialogelemente ausrichten 304
Dialogelemente in Zeilen oder Spalten anordnen 306
Die Größe mehrerer Dialogelemente verändern 307
Die Größe eines Dialogelements verändern 308
Schaltelemente 309
ID-Werte für Aktionsschalter 310
Bildlaufleisten 310
Listenfelder 311
Eingabefelder 313
Statische Darstellung 314
Statische Symbole 316
Kombinationsfenster 317
Benutzerdefinierte Elemente 318
Eigene benutzerdefinierte Dialogelemente erstellen 318
Bibliothek mit benutzerdefinierten Elementen einrichten (.DLL oder .VBX) 319
Benutzerdefinierte Dialogelemente anzeigen 319
Ein benutzerdefiniertes Dialogelement hinzufügen 319
Ein Dialogfenster testen 320
Zwei Dialogfenster anzeigen 320
Dialogeditor anpassen 321

Kapitel 16
Erzeugen von Menüs 325

Menü-Terminologie 325
Aufruf des Menüeditors 326
Neues Menü erzeugen 326
Ein bestehendes Menü bearbeiten 327
Die Arbeitsfläche des Editors 327
Der Dialogfeldausschnitt 327
Der Testmenü-Ausschnitt 327
Der Script-Ausschnitt 328
Menüs bearbeiten 329
Neue Menübefehle einfügen 329
Menüelemente und Separatoren einfügen 330
Menüelemente bearbeiten 330
Benutzen des Dialogfeldausschnitts 330
Menüeintragstext erstellen 332
Eingabe von Eintrags-IDs 333
Menü-Anweisungen verschieben und kopieren 333
Menü-Anweisungen löschen 333
Ad hoc-Menüs erzeugen 334
Menüs testen 334
Bearbeiten von Menüressourcen-Scripts ... 335
Ein Beispielmenü 337
Das Beispielmenü erzeugen 337

Befehle in das Menü einfügen 338
Einfügen von Anweisungen in das Arrange List-Menü 339
Das Menü testen 339

Kapitel 17
Tastenkürzel definieren 341

Tastenkombinationen in der Tastenkürzeltabelle 342
ASCII-Tasten 342
Virtuelle Tasten 342
Den Tastenkürzeleditor aufrufen 343
Neue Tastenkürzeltabelle erzeugen 343
Bestehende Tastenkürzeltabelle bearbeiten 343
Arbeiten mit aktivem Menüeditor 343
Arbeiten mit dem Tastenkürzeleditor 344
Der Script-Ausschnitt 344
Der Dialogfenster-Ausschnitt 344
Eine Tastenkürzeltabelle bearbeiten 345
Ein Tastenkürzel einfügen 345
Ein Tastenkürzel markieren 346
Den Dialogfenster-Ausschnitt benutzen 346
Den Befehlswert eingeben 346
Tastenkürzel angeben 347
Menübefehl hervorheben 347
Prüfen auf doppelte Tastenbelegung 348
Erstellung eines Beispiels für eine Tastenkürzeltabelle 348

Kapitel 18
Erzeugen einer String-Tabelle 351

Der Aufruf des String-Editors 352
Eine neue String-Tabelle erzeugen 352
Eine vorhandene String-Tabelle bearbeiten 352
Das Arbeiten mit String-Tabellen 352
Strings unter Windows 353
Neue Strings eingeben 354
Bearbeiten vorhandener Strings 355
Verändern eines Strings 355
Bearbeiten eines Ressourcen-skripts einer String-Tabelle 355
Verändern des String 355
Erzeugen einer Beispiel String-Tabelle ... 356

Kapitel 19
Anwendung des Paint-Editors 359

Aufruf des Paint-Editors 359
Pixel, Vordergrund- und Hintergrundfarben 360
Anwenden der Werkzeugpalette 360
Markierungsrahmen setzen 361
Schere 362
Vergrößern/Verkleinern 362
Radierer 363
Stift 363
Pinsel 363
Airbrush 363
Farbeimer 364
Linie 365
Text 365
Leere Rahmen zeichnen 365
Ausgefüllte Flächen zeichnen 366
Das Hand-Werkzeug 366
Stilauswahl 367
Arbeiten mit zwei Fensterausschnitten 367
Die Statuszeile lesen 368
Arbeiten mit Farben 368
Die Anzahl der Farben für eine Ressource wählen 369
Vorder- und Hintergrundfarben benutzen 369
Transparente und invertierte Farben ... 370
Transparente und invertierte Farben einstellen 370
Farbpalette verbergen oder anzeigen .. 371
Farben selbst definieren 372
Farben in der Farbpalette bearbeiten ... 372
Palettenindex 372
Eine Farbe bearbeiten 373
Text in eine Bitmap-Zeichnung einfügen .. 373
Text ausrichten 374
Die Auswahl von Schrift, Größe und Textattributen 374
Die Pinselform bestimmen 375

Die Füllmuster bestimmen 375
Die Strichstärke bestimmen 376
Ausrichtung eines markierten Bereichs 377
Skalieren eines markierten Bereichs 377
Allgemeine Optionen des Paint-Editors einstellen . 378
Aktualisierung beider Bilder 378
Mit den Standardfarben des Gerätes speichern 378

Kapitel 20
Symbole erzeugen 379

Neue Symbole erzeugen 379
Hinzufügen eines Symbols zur Projekt Datei . 380
Erzeugen einer seperaten Symbol-Datei . 380
Option Binärdatei 380
Symbol Projektdatei 381
Symbole bearbeiten 381
Ansicht verschiedener Auflösungen . . . 381
Benutzen transparenter oder invertierter Farbbereiche 382
Hinzufügen eines Bildes zu einer Symbolressource . 382
Eigenschaften eines Symbols verändern. . . 383
Anzeigen von Geräteinformationen 384
Erzeugen eines Beispielsymbols 384
Ein Taschenrechnersymbol zeichnen . . . 384
Dreidimensionale Effekte erzeugen 386
Zeichnen der Haushaltsbuchseite 386

Kapitel 21
Mauszeiger erzeugen 389

Einen neuen Mauszeiger erzeugen 389
Einen Mauszeiger der Projektdatei hinzufügen . 390
Eine eigenständige Mauszeigerdatei erzeugen . 390
Datei im Binärformat 390
Mauszeiger-Projektdatei 391
Mauszeiger bearbeiten 391
Farbpalette für Mauszeiger 391
Mit invertierten und durchsichtigen Farben arbeiten 391
Den Kontaktpunkt des Mauszeigers setzen . 392

Kapitel 22
Erstellen von Schriften 393

Schrifttypen . 393
Warum werden Bilder in Schriften abgelegt? . 394
Erstellen einer neuen Schrift-Ressource . . . 395
Zuordnen einer Schrift zu einer Projektdatei . 395
Erstellen einer eigenständigen Schriftdatei . 395
Die Option binäres Format 395
Die Schrift Projektdatei 396
Bearbeiten einer Schrift-Ressource 396
Definition und Hinzufügen von Zeichen zu einer Schrift 396
Festlegen der Schriftgröße 396
Festlegen der Anzahl von Zeichen . 397
Zuordnen des Zeichensatzes 397
Erstellen von Zeichen variabler Breite . 398
Setzen der Breite eines Zeichens oder Bildes . 398
Definieren von Header-Informationen für eine Schrift 399
Ändern von Größe und Attributen 400

Kapitel 23
Erzeugen von benutzerdefinierten Ressourcen . 401

Erzeugen eines Ressourcen-Typs 402
Hinzufügen benutzerdefinierter Ressourcen . 402
Bearbeiten von benutzerdefinierten Ressourcen . 403
Hinzufügen von Ressourcendaten in die Projektdatei 404
Daten ins Ressourcenskript eingeben . . 405

Verwaltung von Daten aus gesonderten Dateien 406
Wie Sie den Ressourcen-Typ RCDATA verwenden . 406

Anhang A
Fehlermeldungen 407

Fehlerklassen . 407
Schwerwiegende Fehler 407
Fehler . 408
Warnungen . 408
Meldungen des Hilfe-Compilers 408
Meldungslisten . 409
Erläuterung der Meldungen 409

Anhang B
Borlands benutzerdefinierte Dialogelemente für Windows 483

Borlands vordefinierte Dialogklasse 483
Borlands vordefinierte Dialogelemente . . . 484
Erweiterungen zu Schaltern und Markierungsfeldern 485
BWCC-Dialogfenster 486
Dialogfenster für Schalter-Eigenschaften 487
Dialogfenster für Schaltfeld-Eigenschaften 488
Dialogfenster für Markierungsfeld-Eigenschaften 488
Dialogfenster für Schattierungs-Eigenschaften 489
Dialogfenster für statischen Text . . . 489
Anpassen bestehender Anwendungen an BWCC . 489
BWCC immer in Windows laden 490
BWCC in C- und C++-Programmen 490
Editierung bestehender Ressourcen . . . 490

Anhang C
Vorkompilierte Header-Dateien 493

Arbeitsweise vorkompilierter Header-Dateien . 493
Nachteile vorkompilierter Header-Dateien 494
Vorkompilierte Header-Dateien verwenden . 494
Die Benennung vorkompilierter Header-Dateien 495
Erforderliche Übereinstimmungen 495
Optimierung vorkompilierter Header-Dateien 496

Anhang D
Verwendung von EasyWin 499

Einfacher Übergang vom Textbildschirm zu WINDOWS . 499
InitEasyWin() . 500
Weitere Funktionen 500

Index . 503

Abbildungen

Abbildung 1.1:
Elemente der Entwicklungsumgebung 31
Abbildung 1.2:
Klassen in einem Programm anzeigen 39
Abbildung 1.3:
Fenster für Symboldeklaration 40
Abbildung 1.4:
Anzeige von globalen Symbolen 41

Abbildung 2.1:
Darstellung eines Projektbaums im Projektfenster 44
Abbildung 2.2:
Das Dialogfenster *TargetExpert* 47
Abbildung 2.3:
Das Dialogfenster *Umgebungsoptionen* ... 56
Abbildung 2.4:
Das Dialogfenster *Hilfsprogramme* 59
Abbildung 2.5:
Das Dialogfeld *Erweiterte Hilfsprogramm-Optionen* 62

Abbildung 3.1:
Das Dialogfenster *Projektoptionen* 64
Abbildung 3.2:
Dialogfenster *Schablonen* 65
Abbildung 3.3:
Das Dialogfenster *Optionshierarchie* 71

Abbildung 4.1:
AppExpert-Dialogfenster *Optionen für die Anwendungserzeugung* 119
Abbildung 4.2:
DRAW.IDE im Projekt-Manager geladen 122

Abbildung 5.1:
ClassExpert 135
Abbildung 5.2:
Dialogfenster *Neue Klasse hinzufügen* 137

Abbildung 6.1
Das Fenster Überwachter Ausdruck 156
Abbildung 6.2
Das Dialogfenster *Darstellung überwachter Ausdrücke* 157
Abbildung 6.3:
Das Fenster Haltepunkte 163
Abbildung 6.4:
Das Dialogfenster Haltepunkt 163

Abbildung 14.1:
Anzeigen der Bezeichner nach Datei geordnet 270

Abbildung 15.1:
Ein typisches Dialogfenster 287
Abbildung 15.2:
Dialogeditor mit leerem Dialogfenster 288
Abbildung 15.3:
Die Werkzeugpalette 294
Abbildung 15.4:
Dialogfenster-Koordinaten 300
Abbildung 15.6:
Dialogfenster Größe der Elemente ändern 307
Abbildung 15.7:
Eingabefeld 313
Abbildung 15.8:
Kombinationsfenster des Dialogfensters Datei Öffnen 317

Abbildung 16.1:
Beispielmenü 337

Abbildung 18.1:
String-Editor mit String-Tabelle und Einträgen . 353
Abbildung 18.2:
String-Editor mit vier definierten Strings . . 358

Abbildung 19.1:
Die Werkzeugpalette des Paint-Editors 361
Abbildung 19.2:
Farbenpalette mit 16 Farben 369
Abbildung 19.3:
Der Farbenindex einer 16-Farben-Palette . 372
Abbildung 19.4:
Text ausrichten. 374

Abbildung 20.1:
Dialogfenster Neues Symbol . 383
Abbildung 20.2:
Vergrößertes Bild des Taschenrechners vor Schattierung . 385
Abbildung 20.3:
Vergrößertes Bild des schattierten Taschenrechners . 386
Abbildung 20.4:
Symbol der Haushaltsbuchführung 387

Abbildung B.1:
Dialogfenster mit Borlands vordefinierten Dialogelementen 484

Tabellen

Tabelle 1.1:
Buchstabensymbole in der Symbolanzeige 41
Tabelle 1.2:
Suchausdrücke der Symbolanzeige 42

Tabelle 3.1:
Verzeichnis der Optionen 79

Tabelle 4.1:
Anwendungsdateien des Beispiels 121

Tabelle 6.1:
Formatspezifikatoren für Debug-Ausdrücke 159
Tabelle 6.2:
Im Fenster *Register* angezeigte CPU-Flags . 162
Tabelle 6.3:
Befehle der Mauspalette des Fensters *Ereignisse* . 167

Tabelle 7.1:
Mausbotschaften . 178
Tabelle 7.2:
Fensterbotschaften 178
Tabelle 7.3:
Eingabebotschaften 178
Tabelle 7.4:
Systembotschaften 178
Tabelle 7.5:
Initialisierungs-botschaften . 179
Tabelle 7.6:
Botschaften der Zwischenablage 179
Tabelle 7.7:
Botschaften für dynamischen Datenaustausch (DDE) 179
Tabelle 7.8:
Non-Client-Botschaften 179
Tabelle 7.9:
Druckerbotschaften 179
Tabelle 7.10:
Steuerbotschaften . 180
Tabelle 7.11:
Stiftbotschaften . 181
Tabelle 7.12:
Multimedia-Botschaften 181
Tabelle 7.13:
Sonstige Botschaften 181
Tabelle 7.14:
Von Microsoft nicht dokumentierte Botschaften . 181

Tabelle 8.1:
Typen von Exceptions 190
Tabelle 8.2:
Optionen für DFA . 196

Tabelle 9.1:
Bibliotheken und Startup-Dateien von Borland . 203
Tabelle 9.2:
32-Bit-Bibliotheken und -Startup-Dateien von Borland 203
Tabelle 9.3:
Optionen für TLINK 204
Tabelle 9.4:
Warnungen von TLINK32 211

Tabelle 10.1
BRCC (Ressourcencompiler von Borland . 222
Tabelle 10.2:
Optionen von RLINK 223
Tabelle 10.3
Optionen von BRC 224
Tabelle 11.1:
IMPLIB-Optionen . 228
Tabelle 11.2: TLIB-Optionen 232

Tabelle 11.3: TLIB-Aktionssymbole 235

Tabelle 12.1:
Optionen für MAKE 240
Tabelle 12.2:
Befehls-Präfixe 246
Tabelle 12.3:
KommandozeilenOperatoren 247
Tabelle 12.4:
Syntaxunterschiede der
Makrodefinition per Kommandozeile
und per Make-Datei 249
Tabelle 12.5:
Vordefinierte Makros 250
Tabelle 12.6:
Weitere vordefinierte Makros 250
Tabelle 12.7:
Makromodifizierer für Dateinamen 251
Tabelle 12.8:
MAKE-Direktiven 251
Tabelle 12.9:
Operatoren in Bedingungsausdrücken 254

Tabelle 14.1:
Speicheroptionen für die Ressource 276
Tabelle 14.2: Bezeichner Präfixe 280

Tabelle 15.1:
Optionen, die in allen Dialogfenstern
Eigenschaften der Dialogelemente
aufgeführt sind. 301
Tabelle 15.2:
Attribute der Elemente 301
Tabelle 15.3:
Ausrichtungsoptionen 305
Abbildung 15.5:
Optionen für die Reihenfolge von
Dialogelementen 307
Tabelle 15.4:
Optionen zur Größenveränderung 308
Tabelle 15.5:
Schaltertypen 309
Tabelle 15.6:
Die vordefinierten Windows Schalter 310
Tabelle 15.7:
Ausrichtungsoptionen 311
Tabelle 15.8:
Die Optionen von Selbstzeichnend 311
Tabelle 15.9:
Listenfenster-Optionen 312
Tabelle 15.10:
Die Optionen des Dialogfensters
Eigenschaften des Eingabefeldes 313
Tabelle 15.11:
Eingabefeld-Optionen für Windows 3.1 ... 314
Tabelle 15.12:
Die Elementtyp-Optionen 315
Tabelle 15.13:
Die Typ-Optionen des
Kombinationsfensters 317
Tabelle 15.14:
Selbstzeichnend-Optionen 317
Tabelle 15.15:
Die Auswahlmöglichkeiten für Attribute
des Kombinationsfensters 318

Tabelle 16.1:
Optionen des Menüs Ansicht 327
Tabelle 16.2.:
Optionen des Menüeditors im
Dialogfeldausschnitt 331

Tabelle 17.1:
Elemente des Dialogfenster-Ausschnitt
im Tastenkürzeleditor 345

Tabelle 19.1: Vergrößern-Befehle 362

Tabelle 22.1:
Optionen zur Größe 397
Tabelle 22.2:
Optionen zum Zeichen 398
Tabelle 22.3:
Schrift-Header-Optionen 399

Tabelle A.1 :
Vordefinierte BWCC-Schalterelemente ... 487
Tabelle A.2 :
Bitmap Offset-Werte 488

Einführung

Borland C++ ist ein leistungsfähiges, professionelles Programmierwerkzeug für die Erstellung und Pflege von DOS-, Win16- und Win32-Anwendungen. Mit seiner integrierten Entwicklungsumgebung und den Kommandozeilen-Hilfsprogrammen unterstützt Borland C++ die Sprachen C und C++.

Dieses Handbuch gliedert sich in drei Teile. Teil I ist eine Einführung in die integrierte Entwicklungsumgebung (Integrated Development Environment, IDE). Teil II beschreibt die Werkzeuge, die für die Erzeugung von Anwendungen aus der Kommandozeile benötigt werden. Teil III zeigt, wie mit dem Resource Workshop-Ressourcen für Windows erstellt werden. Drei Anhänge zu diesem Handbuch erläutern zusätzlich EasyWin (eine Bibliothek, die DOS- in Windows-Programme konvertiert), die vorcompilierten Borland C++ Header-Dateien sowie Fehlermeldungen, die von den Borland C++ Programmierwerkzeugen ausgegeben werden.

Was ist neu in Borland C++?

Borland C++ 4.5 weist zahlreiche Ergänzungen und neue Eigenschaften auf. Es folgt eine kurze Zusammenstellung der wichtigeren Neuheiten und Veränderungen:

- Die Compiler bieten eine verbesserte Code-Erzeugung und raschere Compilierungszeiten.
- Mit ObjectComponents, einer neuen Menge von C++ Klassen, können Sie auf einfache Weise OLE 2.0 in Ihre Anwendungen implementieren. Weitere Informationen hierzu finden Sie im *ObjectWindows Programmierhandbuch*.
- ObjectWindows (OWL) 2.5 enthält neue Klassen, über die Sie durch ObjectComponents Zugriff auf OLE 2.0 erhalten. Weitere Informationen hierzu finden Sie im *ObjectWindows Programmierhandbuch*.

- AppExpert und ClassExpert wurde eine OLE 2.0 Unterstützung hinzugefügt. Weitere Informationen finden Sie in den Kapiteln 4 und 5 dieses Buches.
- AppExpert wurde mit einem Dialog Client-Anwendungsmodell ausgestattet, das ein Dialogfenster als Haupt-Benutzerschnittstelle verwendet. Weitere Informationen finden Sie in Kapitel 4.
- VBX-32s unterstützt Win32s VBX, wodurch Sie auf einfache Weise Ihre VBX-Anwendungen auf 32-Bit-Systeme übertragen können. Weitere Informationen hierzu finden Sie im *ObjectWindows Programmierhandbuch*.
- Mit den neuen 16-Bit-Kommandozeilenoptionen von TLINK können Sie kleinere Windows Programme erstellen. Weitere Informationen finden Sie unter »Weitere Linker-Optionen«.
- Mit OpenHelp können Sie Multi-Online-Hilfedateien schnell und einfach durchsuchen. Nähere Informationen finden Sie unter »Durchsuchen von Multi-Online-Hilfedateien mit OpenHelp«.
- Über WinRun können Sie Windows-Programme in einem DOS-Fenster ausführen. Nähere Informationen finden Sie im *ObjectWindows Programmierhandbuch*.
- Der Turbo Profiler bietet Laufzeitanalyse im 16-Bit-Erweiterungsmodus.

Neu in Version 4.0

Die folgende Liste bietet einen Überblick über die in Version 4.0 neu hinzugekommenen Programmeigenschaften. Sie ist für Anwender gedacht, die ein Upgrade von einer Vorgängerversion von Borland C++ 4.0 durchführen.

- Der 32-Bit-Compiler und die Hilfsprogramme erzeugen 32-Bit-Ziel-Code für Win32s und Windows NT.
- Sie können DOS-Programme unter der Windows-IDE erzeugen.
- Die IDE hat einen grafisch integrierten Debugger zum Austesten von 16-Bit-Windows-Anwendungen.
- Die IDE besitzt einen verbesserten Editor, der es Ihnen ermöglicht, Tastaturmakros aufzuzeichnen, in einem Fenster mehrere Ausschnitte zu bearbeiten und unter Verwendung regulärer Ausdrücke den Text zu durchsuchen. Sie können den Editor zur Bedienung mit Epsilon-, Brief- oder eigenen Tastenbefehlen einrichten.
- Die rechte Maustaste aktiviert lokale Menüs, in denen die mit dem angewählten Objekt verbundenen Befehle aufgeführt sind. Im lokalen Menü aller Editor-Fenster befinden sich beispielsweise häufig benötigte Editierbefehle (die alte Funktionalität der rechten Maustaste erreichen Sie, indem Sie während des Anklickens die Taste [*Strg*] betätigen).

- Die IDE hat eine Multiziel-Projektverwaltung, die Dateiabhängigkeiten optisch darstellt und Ihnen die Verwaltung mehrerer Programme innerhalb eines Projekts ermöglicht.
- Die IDE verfügt zur Darstellung von Klassenbeziehungen über einen neuen Multifenster-Object-Browser.
- Mit dem AppExpert können Sie zügig ObjectWindows 2.0-Windows-Programme erzeugen. ClassExpert hilft Ihnen, Ihre AppExpert-Anwendung zu organisieren und zu verändern.

Hardware- und Software-Anforderungen

Um Borland C++ zu installieren, benötigen Sie folgendes:

- DOS ab Version 4.01
- Windows ab Version 3.1, das im erweiterten Modus für 386-PCs ausgeführt wird
- Einen 80386- oder schnelleren Prozessor (oder kompatibler Prozessor)
- Eine Festplatte mit 40 MB verfügbarem Speicher (für eine vollständige Installation benötigen Sie 100 MB)
- Ein 1,44-MB-Diskettenlaufwerk oder ein CD-ROM-Laufwerk (für die Installation)
- Mindestens 4 MB Erweiterungsspeicher (Extended Memory), um IDE auszuführen (bzw. mindestens 8 MB Erweiterungsspeicher bei einem 386er-System)
- Eine Windows-kompatible Maus

Die folgenden Einrichtungen sind zwar nicht erforderlich, können aber die Leistung von Borland C++ beträchtlich erhöhen:

- 8 MB RAM (oder mehr)
- Ein mathematischer Koprozessor 80x87 (für Programme mit Gleitkomma-Arithmetik). Borland C++ emuliert einen Arithmetikprozessor, falls Sie keinen zur Verfügung haben.

Konventionen

Dieses Handbuch benutzt folgende Schriftarten und Symbole:

`Letter Gothic` In der Schriftart Letter Gothic erscheinen Programm-Listings, Eingaben des Benutzers oder Ausgaben von Beispielprogrammen.

Kursiv — Namen von C++ Funktionen, Klassennamen und Strukturnamen werden kursiv geschrieben, um wichtige Worte hervorzuheben und auf Namen von Variablen hinzuweisen (Bezeichner).

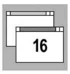
Fettschrift — Fettschrift kennzeichnet außerhalb von Programmlistings reservierte Schlüsselwörter von Borland C++, wie **char**, **switch**, **near**, **cdecl** , **#include** usw. Auch Kommandozeilenoptionen (**-AK**) werden fett dargestellt.

[*EINGABETASTE***]** — In dieser Darstellung erscheinen bestimmte Tasten, die Sie betätigen müssen, z.B. »Ein Druck auf [*Esc*] entfernt das Menü vom Bildschirm«.

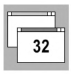
[*TASTE 1*][*TASTE 2***]** — Tastenkombinationen, bei denen mehrere Tasten gleichzeitig betätigt werden müssen, erscheinen als [*TASTE 1*][*TASTE 2*]. Sie können zum Beispiel den Befehl *Bearbeiten/Widerrufen* auslösen, indem Sie die Tasten [*Strg*] und [*Umschalt*] halten und [*Z*] drücken. Diese Tastenkombination wird durch [*Strg*][*Umschalt*][*Z*] wiedergegeben.

GROSSCHRIFT — Dies ist die Darstellung für Verzeichnisse, Dateinamen und Anwendungen.

Menü/Auswahl — Anstelle von »Wählen Sie den Befehl *Speichern* aus dem Menü *Datei*« wird in den Handbüchern von Borland die Konvention »Wählen Sie *Menü/Speichern*« verwendet.



16-Bit-Windows



32-Bit-Windows

Hilfesystem

Über das Online-Hilfesystem haben Sie jederzeit Zugriff auf ausführliche Informationen von Borland C++. Die meisten Produktinformationen finden Sie sowohl dort als auch in den Handbüchern. Folgende Themen werden jedoch nur in der Online-Hilfe behandelt:

- IDE-Menübefehle
- Editor KEYMAPPER
- Beispiel-Code der Laufzeitbibliothek
- Windows API

Die Online-Hilfe können Sie folgendermaßen aufrufen:

- In der IDE wählen Sie das Menü *Hilfe* aus oder drücken Sie [*F1*].
- In den Dialogfenstern klicken Sie auf den Schalter *Hilfe* oder drücken Sie [*F1*].
- Für Hilfe zu Menübefehlen markieren Sie den Befehl und drücken Sie [*F1*].

Durchsuchen mehrerer Hilfedateien mit OpenHelp

OpenHelp ist ein Dienstprogramm, mit dem Sie mehrere Online-Hilfedateien von einem einzelnen Dialogfenster aus durchsuchen können. Wenn Sie nach einem bestimmten Stichwort suchen, durchsucht OpenHelp die Stichwortverzeichnisse sämtlicher von Ihnen angegebenen Hilfedateien, d.h. Sie verfügen nun mit einem einzigen Befehl über sämtliche Windows-Hilfedateien.

OpenHelp kann auf verschiedene Weise gestartet werden:

- Klicken Sie auf den Schalter *Alles durchsuchen* in der Hilfeschalterleiste jeder beliebigen Borland Hilfedatei.
- Wählen Sie *Hilfe/Stichwortsuche* in der Borland C++ IDE (oder drücken Sie [*F1*]).
- Doppelklicken Sie auf das OpenHelp-Symbol im Borland C++ Gruppenfenster auf der Windows-Arbeitsfläche.

Wenn Sie auf OpenHelp zugreifen, können Sie die von Ihnen angegebenen Hilfedateien durchsuchen, um die benötigten Informationen aufzufinden. OpenHelp durchsucht standardmäßig eine vordefinierte Menge von Hilfedateien, die als *Suchbereich* bezeichnet wird. Sie können OpenHelp an Ihre Bedürfnisse anpassen, indem Sie neue Suchbereiche hinzufügen und bereits vorhandene modifizieren.

Für weitere Informationen zu OpenHelp und Anweisungen zum Anpassen der Suchbereiche drücken Sie (wenn Sie sich in Borland C++ befinden) die Taste [*F1*] und suchen das Thema *OpenHelp verwenden*.

Wie erreichen Sie Borland?

Bei Problemen mit Borland C++ bitten wir Sie, zunächst die Handbücher sowie die Datei README sorgfältig durchzulesen.

Wenn danach Fragen offenbleiben, können Sie sich an eine der folgenden Stellen wenden:

1. Fragen Sie Ihren Software-Händler oder einen Benutzer-Club
2. Rufen Sie das Borland Informationssystem (kurz BORIS) an. Sie erhalten hier rund um die Uhr technische Informationen, Tips & Tricks sowie Antworten auf die im technischen Support am häufigsten gestellten Fragen.

 BORIS ist ein Fax-Server, auf dem gesprochene Informationen gespeichert sind. Diese Informationen können Sie auch auf Ihr Telefax schicken lassen.

Sie erreichen BORIS (vorausgesetzt, Sie verfügen über ein Mehrfrequenzwahl-Telefon), unter folgender Telefonnummer:

(061 03) 979-225

Nähere Informationen erhalten Sie von unserem Kundendienst oder online von BORIS.

3. Schreiben Sie an:

 Borland GmbH
 C/C++ Technischer Support
 Monzastr. 4c
 63225 Langen

4. Informationen können Sie auch aus unserem Borland Net beziehen. Sie können dort direkt mit unseren Hotline-Spezialisten und anderen Anwendern kommunizieren und unter anderem folgendes abrufen:

 - Aktuelle Software und Utilities
 - Treiber, Patches und Beispielprogramme
 - Tips und Tricks zu Borland C++ und anderen Borland Produkten
 - und vieles mehr.

 Sie erreichen Borland Net unter folgender Telefonnummer:

 (06103) 979-353

 Beachten Sie folgende Übertragungsparameter:

 Übertragungsrate 2400 bis 9600 Baud
 8 Datenbits
 No Parity
 1 Stopp-Bit

 Das Borland Net ist kostenpflichtig und nur für eingetragene Anwender in vollem Umfang nutzbar. Nähere Informationen erhalten Sie von unserem Kundendienst oder online im Borland Net.

5. Geben Sie im CompuServe Mail-System GO BORGER ein. Im Forum Borland Deutschland wählen Sie dann C/C++ aus dem Menü aus. Unter C/C++ finden Sie Bibliotheken mit technischen Informationen und Antworten auf gängige Fragen. Die Benutzung von CompuServe ist kostenpflichtig. Wenn Sie Informationen wünschen, wenden Sie sich bitte an folgende Anschrift:

 CompuServe GmbH
 Jahnstr. 2
 82008 Unterhaching bei München

6. Senden Sie ein Fax mit Ihrer Problemstellung oder Ihren Fragen an folgende Nummer:

 (061 03) 979-296

7. Rufen Sie den technischen Support der Borland GmbH unter folgender Nummer zu folgenden Zeiten an:

 Dienstag bis Donnerstag 14.00 bis 17.00

 Rufnummern:

 (061 03) 979-342 für C/C++ und Pascal

 (061 03) 979-343 für C/C++ und Pascal

Sie sollten in allen Fällen die Handbücher sowie die folgenden Informationen zur Hand haben:

1. Die Versionsnummer von Borland C++. (Diese erhalten Sie, wenn Sie im Hauptmenü der IDE das *Hilfe*-Menü öffnen und *Info über Borland C++...* wählen.)
2. Produktnamen und Modellbezeichnung Ihres Computers sowie die Namen eventueller Zusatz-Hardware – insbesondere von Grafikkarten.
3. Name und Versionsnummer des Betriebssystems (PC-DOS oder MS-DOS). Diese Informationen erhalten Sie durch Eingabe des Befehls VER auf der DOS-Kommandoebene.
4. Die Nummer Ihrer Windows-Version.

T E I L I

Verwenden der integrierten Entwicklungsumgebung

Teil I dieses Handbuches beschreibt, wie Sie die Bestandteile des AppExpert der integrierten Entwicklungsumgebung (Integrated Development Environment, IDE) von Borland C++ verwenden.

Die IDE integriert die Entwicklung von Anwendungen für DOS, Win16 und Win32. Mit dem Projekt-Manager können Sie auf einfache Weise unterschiedliche Anwendungstypen aus einer einzigen Projektdatei erzeugen. Über AppExpert und ClassExpert haben Sie die Möglichkeit, ObjectWindows 2.0 zu nutzen. Untersuchen Sie Ihren Quelltext mit dem integrierten Debugger auf Fehler und sehen Sie mit dem ebenfalls integrierten Browser Klassenobjekte und -hierarchien durch, ohne die IDE zu verlassen.

Folgende Kapitel behandeln die über die IDE verfügbaren Dienstprogramme:

- Kapitel 1 bietet Ihnen eine Einführung in die IDE von Borland C++ und führt Sie durch die Generierung einfacher DOS-, Windows- und 32-Bit-Windows-Programme.
- Kapitel 2 beschreibt den Projekt-Manager und erläutert die Verwendung von TargetExpert und Quelltextsammlungen beim Erstellen der Projekte für Ihre Anwendungen.
- Kapitel 3 zeigt, wie Sie Schablonen und überschreibende lokale Optionen verwenden, um Projektoptionen zu setzen, und wie Sie von der IDE aus compilieren. Zusätzlich enthält dieses Kapitel eine vollständige Beschreibung aller für die IDE und die Kommandozeilen-Hilfsprogramme verfügbaren Optionen.

- Kapitel 4 enthält eine Beschreibung von AppExpert und erläutert, wie die Quelltext-Grundlage für ObjectWindows 2.0-Anwendungen erzeugt wird.
- Kapitel 5 beschreibt, wie Sie Ihre Anwendungen mit Hilfe von ClassExpert modifizieren.
- Kapitel 6 zeigt die Verwendung des integrierten Debuggers, mit dem Sie Ihren Programm-Code schrittweise durchlaufen können.

K A P I T E L 1

Erste Schritte

Borland C++ ist ein Entwicklungspaket mit Dienstprogrammen für Windows, Kommandozeilen-Dienstprogrammen und Bibliotheken, mit denen Sie Programme für DOS, Windows, WIN32s und Windows NT entwickeln können. Dieses Kapitel enthält eine Arbeitsbeschreibung von Borland C++ – der IDE (Entwicklungsumgebung), der Projektverwaltung, AppExpert sowie der Dienst- und Hilfsprogramme.

In diesem Kapitel erhalten Sie Informationen über

- die Installation und Konfigurierung von Borland C++
- die Arbeit mit dem Editor
- die Verwendung von Syntaxhervorhebung für eigene Programme
- die Verwendung der Mauspalette
- die Verwendung des Meldungsfensters
- die Durchsicht des Programms
- den Aufruf und die Verwendung anderer Dienstprogramme von der IDE (Entwicklungsumgebung) aus

Die anderen Hilfsprogramme und die Dienstprogramme für die Kommandozeile werden in diesem Kapitel ebenfalls kurz beschrieben.

Borland C++ installieren

Borland C++ enthält sowohl DOS- als auch Windows-Anwendungen. Stellen Sie vor Installationsbeginn sicher, daß Ihr Computer die Mindestanforderungen an die Hardware und Software erfüllt.

Hardware- und Softwareanforderungen

Um Borland C++ zu installieren, benötigen Sie folgendes:

- DOS ab Version 4.01
- Windows ab Version 3.1, das im erweiterten Modus für 386-PCs ausgeführt wird
- Eine Festplatte mit 40 MB verfügbarem Speicher (für eine vollständige Installation benötigen Sie 80 MB)
- Ein 1,44-MB-Diskettenlaufwerk oder ein CD-ROM-Laufwerk (für die Installation)
- Mindestens 4 MB Erweiterungsspeicher (Extended Memory)
- Eine Windows-kompatible Maus

Die folgenden Einrichtungen sind zwar nicht erforderlich, können aber die Leistung Ihres Computers beträchtlich erhöhen:

- 8 MB RAM
- Ein mathematischer Koprozessor 80x87 (für Programme mit Gleitkomma-Arithmetik). Borland C++ emuliert einen Arithmetikprozessor, falls Ihnen keiner zur Verfügung steht.

Installationsschritte

Das Installationsprogramm installiert Borland C++ (die IDE, die Kommandozeilen-Dienstprogramme, ObjectWindows und den Turbo-Debugger) sowie Win32s. (Mit Win32s können Sie 32-Bit-Programme unter dem 16-Bit-Windows ausführen.) Das Installationsprogramm läuft unter Windows, Win32s, Windows und Windows NT; unter Windows NT können jedoch nicht alle Programme ausgeführt werden.

Wenn Sie ein Hilfsprogramm zur Plattenkomprimierung verwenden, sollten Sie vor der Installation von Borland C++ die Datei INSTALL.TXT lesen.

Überprüfen Sie vor Installationsbeginn, ob Ihr Computer die Hardware- und Softwareanforderungen für die Installation erfüllt. Weitere Informationen zur Installation von Borland C++ können Sie der Datei INSTALL.TXT auf Diskette 1 entnehmen. (Da diese Datei nicht komprimiert ist, kann sie mit einem beliebigen Editor aufgerufen werden.)

Die Anweisungen für die Installation von Diskette und CD-ROM sind weitgehend identisch. Sie sollten jedoch trotzdem die Datei INSTALL.TXT oder die Bemerkungen auf der CD-ROM lesen.

Zur Installation von Borland C++ gehen Sie wie folgt vor:

1. Legen Sie Diskette 1 in das Diskettenlaufwerk ein (normalerweise A oder B).

2. Rufen Sie Windows auf und wählen Sie im Programm-Manager *Datei/Ausführen*.

3. Geben Sie je nach Bezeichnung Ihres Diskettenlaufwerks a:\installaus oder b:\installaus ein, und drücken Sie die Eingabetaste. Daraufhin erscheint das Dialogfeld für die Installation. Am unteren Rand dieses Dialogfeldes wird angezeigt, wieviel Speicherplatz für eine vollständige Installation auf der Festplatte erforderlich ist (Zielanforderungen). Außerdem wird der gesamte verfügbare Speicherplatz angezeigt. Bevor Sie mit der Installation fortfahren, überprüfen Sie, ob genügend Speicherplatz vorhanden ist. Lesen Sie die Datei INSTALL.TXT, wenn Ihr Computer mit einer Plattenkomprimierung arbeitet, da dann möglicherweise mehr Speicherplatz benötigt wird, als aufgeführt ist.

4. Möchten Sie nur bestimmte Dateien installieren, klicken Sie den Schalter *Borland C++ 4.0 konfigurieren* an. Daraufhin wird ein weiteres Dialogfeld mit Schaltern und Beschreibungen für die einzelnen Bereiche von Borland C++ angezeigt. Klicken Sie den Schalter des Bereichs an, den Sie konfigurieren wollen. Für den ausgewählten Bereich wird ein weiteres Dialogfeld geöffnet. Bei der Standardinstallation werden alle Dateien installiert. Löschen Sie die Markierungen zu den Dateien, die Sie nicht installieren möchten. Klicken Sie nun OK an, und wiederholen Sie diesen Vorgang für jeden ausgewählten Bereich. Klicken Sie erneut auf OK, um in das erste Dialogfeld für die Installation zurückzukehren.

5. Das Installationsprogramm zeigt die Standardverzeichnisse an, in denen die Dateien installiert werden. Möchten Sie Borland C++ in einem anderen Verzeichnis installieren, geben Sie den Namen des Verzeichnisses ein.

 - **Das Borland C++ Zielverzeichnis** ist das Hauptverzeichnis, unter dem alle anderen Dateien installiert werden. Dieses Verzeichnis wird standardmäßig mit C:\BC4 bezeichnet.
 - **Das Borland C++ Arbeitsverzeichnis** ist das Verzeichnis, in dem bei der Installation die Borland C++ Anwendungen gespeichert werden. Normalerweise wird dieses Verzeichnis mit C:\BC4\BIN bezeichnet.

6. Das Installationsprogramm erzeugt standardmäßig eine Windows-Programmgruppe mit allen Borland C++ Symbolen. Soll diese Gruppe nicht erzeugt werden, entfernen Sie die Markierung von der Option *Borland C++ Gruppe erzeugen*.

7. Win32s wird ebenfalls standardmäßig installiert. Soll Win32s nicht installiert werden, wählen Sie diese Option ab.

8. Wählen Sie *LAN Windows Konfiguration*, wenn Windows auf Ihrem Computer im Netzwerk installiert ist.
9. Klicken Sie Installieren an, um mit dem Kopiervorgang zu beginnen. Lesen Sie nach Beendigung der Installation die Datei README.TXT. Diese Datei enthält die letzten Änderungen des Produkts, der Dokumentation und der Online-Hilfe. Eine Beschreibung aller bei der Installation erzeugten Symbole finden Sie in der Datei INSTALL.TXT auf Diskette 1 oder im Verzeichnis C:\BC4.

Überprüfen Sie nach Beendigung der Installation, ob die Einträge **FILES** und **BUFFERS** in der Datei CONFIG.SYS mindestens auf 40 gesetzt sind. Informationen zur Datei CONFIG.SYS finden Sie in Ihrer DOS-Dokumentation.

Während der Installation werden die folgenden Dateien geändert:

- Die Datei AUTOEXEC.BAT enthält nun auch den Pfad für Borland C++ (standardmäßig **C:\BC4\BIN**).
- Die Datei WIN.INI enthält nun auch den Abschnitt [BCW4.0 INSTALL], den TASM zur Lokalisierung von Borland C++ verwendet (wenn TASM installiert wird). Außerdem wird die Erweiterung .IDE im Abschnitt **[EXTENSIONS]** aus der IDE (Datei BCW.EXE) zugeordnet.
- Die Datei SYSTEM.INI enthält zwei Gerätetreiber:

```
device = c:\bc4\bin\tddebug.386
device = c:\bc4\windpmi.386
```

- Wenn Sie über Windows NT verfügen, wird der Eintrag **NTCMDPROMPT** zu der Datei CONFIG.NT hinzugefügt.

Die Datei FILELIST.TXT enthält eine Liste aller Dateien, die mit Borland C++ geliefert werden. Müssen Sie Borland C++ Dateien löschen, da Sie freien Festplattenplatz benötigen, sollten Sie zuerst diese Datei lesen.

Borland C++ starten

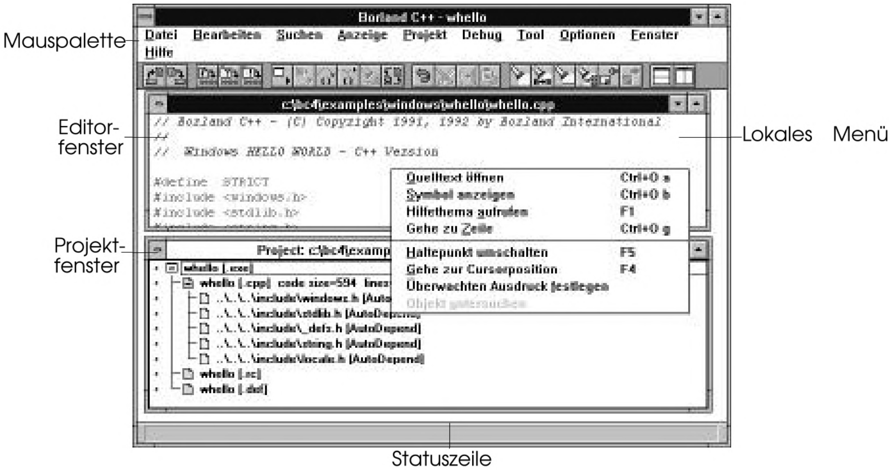


Doppelklicken Sie zum Aufruf der IDE auf das Borland C++ Symbol von Windows (das links dargestellt ist). In der IDE können Sie Programmierprojekte entwerfen, bearbeiten, compilieren, linken, debuggen und verwalten. Die IDE besteht aus folgenden Komponenten:

- Einem Editor und einer Symbolanzeige, die in diesem Kapitel beschrieben werden;
- Einer Projektverwaltung, die in Kapitel 2 beschrieben wird
- Einem Debugger, der in Kapitel 6 beschrieben wird

Abbildung 1.1 zeigt einige Komponenten der Entwicklungsumgebung.

Abbildung 1.1: Elemente der Entwicklungsumgebung



Die IDE enthält kontextbezogene lokale Menüs, in denen Sie Objekte schnell bearbeiten können. Zum Aufrufen eines lokalen Menüs klicken Sie ein Fenster mit der rechten Maustaste an, das Sie zuerst durch Drücken der linken Maustaste auswählen müssen, oder klicken Sie mit der rechten Maustaste eine Option in einem Fenster an, oder drücken Sie [*Alt*]+[*F10*]. (Das lokale Menü ändert sich abhängig von der Auswahl, die Sie getroffen haben.) Möchten Sie beispielsweise in eine Zeile in einem Editorfenster springen, klicken Sie das Editorfenster mit der rechten Maustaste an, wählen Sie *Gehe zu Zeile* aus, und geben Sie die Nummer der Zeile ein, die Sie angezeigt haben möchten. Die Option *Gehe zu Zeile* kann nur aufgerufen werden, wenn Sie zuvor ein Editorfenster ausgewählt haben. Wenn Sie ein lokales Menü in einem Projektfenster öffnen, werden andere Menüpunkte angezeigt.

Auch der Aufbau der Mauspalette hängt davon ab, welches Fenster Sie auswählen. Für den Editor, die Symbolanzeige, den Debugger, die Projektverwaltung, das Meldungsfenster, die Arbeitsfläche und den ClassExpert steht eine konfigurierbare Mauspalette zur Verfügung. Wenn Sie den Mauszeiger auf einen Schalter der Mauspalette bewegen, wird in der Statuszeile am unteren Rand der IDE eine Kurzbeschreibung des Schalters angezeigt.

Wird ein Schalter der Mauspalette blaß dargestellt, weist dies darauf hin, daß Ihnen der Befehl, den dieser Schalter repräsentiert, im aktuellen Zusammenhang nicht zur Verfügung steht. So wird zum Beispiel in einem geöffneten Editorfenster der Schalter *Text aus Zwischenablage* einfügen blaß dargestellt, wenn die Zwischenablage keinen Text enthält.

Hilfe aufrufen

Durch das Borland C++ Hilfesystem haben Sie jederzeit Zugriff auf ausführliche Informationen über Borland C++. Den größten Teil der Produktinformationen finden Sie sowohl in den Handbüchern als auch in der Online-Hilfe. Die folgenden Themen sind jedoch nur in der Online-Hilfe enthalten:

- IDE-Menübefehle
- Editor KEYMAPPER
- Beispielcode der Laufzeitbibliothek
- Windows-API

Die Online-Hilfe können Sie wie folgt aufrufen:

- In der IDE wählen Sie das Menü *Hilfe* aus oder drücken die Taste [*F1*].
- In den Dialogfenstern klicken Sie den Hilfe-Schalter an oder drücken die Taste [*F1*].
- Für Menübefehle wählen Sie den Menübefehl aus und drücken die Taste [*F1*].

Die IDE konfigurieren

Sie können die IDE für die automatische Ausführung von Tasks, wie z.B. der Erstellung einer Sicherungskopie von Dateien in Editorfenstern oder der Behandlung von Ereignissen, konfigurieren. In diesem Abschnitt werden die Konfigurationsmöglichkeiten näher beschrieben.

Im Dialogfenster *IDE-Einstellungen* können Sie den Editor, die Symbolanzeige, den Debugger, die Projektfenster und andere Elemente der IDE konfigurieren. Die Optionen sind in der Datei BCCONFIG.BCW gespeichert.

Wählen Sie zum Öffnen des Dialogfensters *IDE-Einstellungen* den Menüpunkt *Optionen/IDE* aus. Das daraufhin angezeigte Dialogfenster enthält in der linken Hälfte eine Liste der verfügbaren Optionen.

Sie können die Liste der Themen durch Drücken der Tasten [+] oder [–] erweitern oder einschränken. Einige Themen sind in Unterthemen untergliedert. So ist beispielsweise das Thema Editor in die Unterthemen Optionen, Datei und Anzeige unterteilt. Enthält ein Thema Unterthemen, die nicht angezeigt werden, ist dieses Thema durch ein Pluszeichen (+) neben dem Themennamen gekennzeichnet. Klicken Sie dieses Pluszeichen an, werden die Unterthemen ebenfalls aufgelistet, und das Pluszeichen wird zu einem Minuszeichen (–). Wenn Sie das Minuszeichen anklicken, wird die Liste mit den Unterthemen wieder ausgeblendet. Themen, die nicht in Unterthemen untergliedert sind, sind durch einen Punkt neben dem The-

mennamen gekennzeichnet. Wenn Sie ein Thema anklicken, werden seine Merkmale in der rechten Hälfte des Dialogfensters angezeigt.

In diesem Kapitel werden nicht alle Themen des Menüs *Optionen/IDE* behandelt. Umfangreiches Referenzmaterial zu allen Themen und Optionen finden Sie in der Online-Hilfe. Zum Aufrufen der Online-Hilfe klicken Sie den Hilfe-Schalter im Dialogfenster an.

Einige Themen im Zusammenhang mit Tasks oder Teilen der IDE werden in diesem Handbuch an anderer Stelle behandelt. Einträge zu speziellen Themen finden Sie im Index dieses Handbuchs.

Mauspaletten ändern

Die IDE stellt Mauspaletten für Editor-, Symbolanzeige-, Debugger-, Projekt-, Meldungs-, Arbeitsbereichs- und ClassExpert-Fenster zur Verfügung. Die entsprechende Mauspalette wird angezeigt, wenn Sie einen dieser Fenstertypen auswählen. Sie können die Mauspaletten so konfigurieren, daß sie nur die Schalter enthalten, die Sie benötigen.

Mit den folgenden Schritten können Sie Schalter hinzufügen oder entfernen:

1. Wählen Sie im Hauptmenü den Menüpunkt *Optionen/IDE* aus.
2. Wählen Sie das Thema *Mauspalette* auf der linken Seite aus. In der rechten Hälfte des Dialogfensters werden allgemeine Optionen für alle Mauspaletten angezeigt.

Über die hier angezeigten Optionen können Sie die Position der Mauspalette am oberen oder unteren Rand der IDE und ihr Aussehen festlegen. Kurze Hilfeinformationen dazu erhalten Sie in der Statuszeile, wenn Sie den Mauszeiger auf einen Schalter bewegen (Option *Hilfetext in Statuszeile* markieren).

3. Wählen Sie das Thema *Konfigurieren* unter *Mauspalette* aus. Die Optionen auf der rechten Seite enthalten Informationen zu den Mauspaletten.
4. Wählen Sie den Mauspalettentyp aus, den Sie bearbeiten möchten (Editor, Symbolanzeige, Debugger, Projekt, Meldung, Desktop oder ClassExpert).

Die linke Spalte (*Verfügbare Schalter*) enthält alle verfügbaren unbenutzten Schalter und die dazugehörigen Funktionsbeschreibungen. Die rechte Spalte (*Aktive Schalter*) enthält nur die Schalter für die ausgewählte Mauspalette.

5. Zum Hinzufügen eines Schalters doppelklicken Sie auf das betreffende Symbol in der Liste der verfügbaren Schalter oder wählen es aus und klicken den nach rechts zeigenden Pfeil an. Der Schalter wird dann in die Liste der aktiven Schalter aufgenommen.
6. Zum Entfernen eines Schalters doppelklicken Sie auf das betreffende Symbol in der Liste der aktiven Schalter oder wählen es aus und klicken den nach links zeigenden Pfeil an. Der Schalter wird dann in die Liste der verfügbaren Schalter eingefügt.

Mit den nach oben und nach unten zeigenden Pfeilen können Sie die Anordnung der Schalter in der Mauspalette verändern. Der ausgewählte aktive Schalter wird in der Liste nach oben oder nach unten verschoben. Der oberste Schalter erscheint am äußersten linken Rand der Mauspalette, der unterste Schalter ganz rechts.

Es besteht auch die Möglichkeit, alle Mauspaletten gleich zu gestalten. Dazu müssen Sie zuerst eine Mauspalette in der Fensterliste auswählen und dann den Schalter *Layout kopieren* aktivieren. Danach wird ein Dialogfenster geöffnet, in dem Sie alle Mauspaletten markieren können, die Sie so wie die ausgewählte Mauspalette gestalten möchten. Wenn Sie beispielsweise zuerst die Mauspalette des Editors auswählen und dann auf *Layout kopieren* klicken, wird der Menüpunkt *Editor* im sich öffnenden Dialogfenster blaß dargestellt. Wenn Sie daraufhin *Projekt* und *Meldung* markieren, erhalten diese Mauspaletten den gleichen Aufbau wie die des Editors.

Möchten Sie eine Mauspalette auf ihre ursprünglichen Standardwerte zurücksetzen, wählen Sie die Mauspalette aus der Liste aus und klicken dann den Schalter *Layout wiederherstellen* an.

Sie können die Abstände zwischen den Schaltern der Mauspaletten mit Trennzeichen verändern. Wählen Sie dazu einen Schalter aus der Liste der aktiven Schalter aus. Klicken Sie dann den Schalter *Trennzeichen* an. Das Trennzeichen wird vor dem ausgewählten Schalter eingefügt.

IDE-Vorgaben setzen

Über diese Vorgaben können Sie die automatische Speicherung und die Funktionsweise von Fenstern festlegen.

Zum Setzen der Vorgaben gehen Sie wie folgt vor:

1. Wählen Sie *Optionen/IDE/Vorgaben* aus.
2. Markieren Sie die gewünschten Optionen. Informationen zu den einzelnen Optionen finden Sie in der Online-Hilfe.
3. Wählen Sie OK aus.

IDE-Vorgaben speichern

Die IDE speichert automatisch einige Informationen, wenn Sie die IDE verlassen, ein Projekt erzeugen oder das Hilfsprogramm MAKE ausführen, mit einem Transfer-Dienstprogramm arbeiten, den integrierten Debugger starten oder ein Projekt schließen, öffnen oder speichern. Die automatische Speicherung können Sie über *Optionen/IDE/Vorgaben* steuern. Um die Einstellungen manuell zu speichern, gehen Sie wie folgt vor:

1. Wählen Sie *Optionen/Speichern* aus.
2. Markieren Sie die Option *Oberfläche*, um die Einstellungen in den Abschnitten Editor, Syntax-Hervorhebung, Mauspalette, Symbolanzeige und Vorgaben im Dialogfenster *IDE-Einstellungen* zu speichern. Die Einstellungen werden in der Datei BCCONFIG.BCW abgespeichert.
3. Markieren Sie die Option *Desktop*, um die Informationen für geöffnete Fenster und deren Positionen zu sichern. Diese Daten werden in der Datei *Projektname*<*prjname*>.DSW gespeichert. Ist kein Projekt geöffnet, werden die Daten in die Datei BCWDEF.DSW geschrieben.
4. Markieren Sie die Option *Projekt*, um Änderungen in der Datei PROJEKT(.IDE) einschließlich der Build-Optionen und der Knotenattribute zu speichern.

Einsatz des Editors

In der IDE können Sie bis zu 32 Editorfenster öffnen.

In den Editorfenstern erzeugen und bearbeiten Sie Ihren Programmcode. Wenn Sie eine Datei editieren, werden in der IDE-Statuszeile die Zeilennummer und die Koordinaten des Cursors angezeigt. Steht der Cursor beispielsweise in Zeile 1 des Editorfensters auf der ersten Zeichenposition, wird in der Statuszeile der Wert 1:1 angezeigt. Steht der Cursor in Zeile 68 auf der Zeichenposition 23, wird der Wert 68:23 angezeigt. In der IDE-Statuszeile wird auch angezeigt, ob sich der Cursor im Überschreib- oder Einfügemodus befindet. (Zwischen diesen beiden Modi kann mit der Taste [*Einfg*] umgeschaltet werden.) Außerdem wird das Wort *Geändert* angezeigt, wenn die Datei im ausgewählten Fenster geändert wurde.

Mit dem Editor können Sie mehrere Editierschritte rückgängig machen, wenn Sie *Bearbeiten/Rückgängig* auswählen oder die Tastenkombination [*Alt*]+[*Rücktaste*] drücken. Haben Sie beispielsweise eine Textzeile gelöscht und danach Text eingefügt, wird zuerst die Einfügung – der letzte Editierschritt – und erst danach das Löschen zurückgenommen. Sie können festlegen, wieviele Editierschritte zurückgenommen werden können, indem Sie *Optionen/IDE/Editor/Optionen* auswählen und die maximale Anzahl der rückgängig machbaren Aktionen festlegen.

IDE-Editor konfigurieren

Sie können den Editor nach dem Vorbild anderer Editoren (z.B. Brief oder Epsilon) konfigurieren. Der IDE-Editor arbeitet mit Tastenbelegungsdateien (.CKB-Dateien). Diese Dateien enthalten die Tastenkürzel für den Editor. Diese Tastenbelegungen gelten auch für andere Fenster.

Die .CKB-Dateien konfigurieren auch andere Fenster in der IDE. Weitere Informationen hierzu finden Sie in der Online-Hilfe.

Es stehen vier Standarddateien mit der Erweiterung .CKB zur Verfügung. Wählen Sie zum Aufrufen einer dieser Dateien *Optionen/IDE/Editor* aus und klicken Sie eine der Schnelleinstellungen (Standard-Tastaturbelegung, IDE klassisch, BRIEF-Emulation oder Epsilon-Emulation). Anweisungen zum Editieren oder Erzeugen eigener .CKB-Dateien finden Sie in der Online-Hilfe (unter dem Stichwort Keymapper).

Syntaxhervorhebung

Mit der Syntaxhervorhebung können Sie Farb- und Schriftartattribute (wie z.B. Fettschrift) für bestimmte Programmelemente definieren. So können Sie sich beispielsweise Kommentare in Blau und Strings in Rot anzeigen lassen. Die Syntaxhervorhebung ist standardmäßig aktiviert. Zum Deaktivieren der Syntaxhervorhebung gehen Sie wie folgt vor:

1. Wählen Sie *Optionen/IDE/Syntax-Hervorhebung* aus.
2. Heben Sie die Markierung von *Syntaxhervorhebung verwenden* auf.

Die Syntaxhervorhebung gilt für die Dateien, deren Erweiterungen in der Liste der Syntaxerweiterungen aufgeführt sind (standardmäßig die Erweiterungen .CPP, .C, .H und .HPP). Sie können Erweiterungen beliebig hinzufügen oder löschen, müssen sie jedoch durch Semikolons voneinander trennen.

Im Abschnitt Syntaxhervorhebung stehen Ihnen das Standardfarbschema und vier vordefinierte Farbeinstellungen (Schalter) zur Verfügung.

Zur Auswahl eines vordefinierten Farbschemas gehen Sie wie folgt vor:

1. Wählen Sie *Optionen/IDE/Syntax-Hervorhebung* aus.
2. Wählen Sie eines der vier Farbschemata aus, indem Sie den entsprechenden Schalter anklicken; der Beispieltext übernimmt dann das von Ihnen ausgewählte Farbschema. Sie können ein Farbschema als Basis für die Festlegung der Syntaxhervorhebung verwenden.

Zur manuellen Auswahl der Farben für die Syntaxhervorhebung gehen Sie wie folgt vor:

1. Wählen Sie die Option *Optionen/IDE/Syntax-Hervorhebung/Konfigurieren* aus. Elemente und der Beispieltext werden rechts im Dialogfenster *IDE-Einstellungen* angezeigt.

2. Wählen Sie aus der Elementliste ein Element aus, das Sie ändern möchten (z.B. Kommentar), oder klicken Sie das Element im Beispieltext an. Dadurch wird das Element in der Elementliste ausgewählt. Möglicherweise müssen Sie in der Liste blättern, um weitere Elemente anzuzeigen. Der Beispieltext erscheint in der Schriftart, die Sie in *Editor/Anzeige* des Dialogfensters *IDE-Einstellungen* ausgewählt haben.
3. Wählen Sie eine Farbe für das Element aus. Das Element im Beispieltext wird danach in der von Ihnen ausgewählten Farbe angezeigt. Durch Klicken mit der linken Maustaste wählen Sie eine Vordergrundfarbe für das Element aus. Diese Auswahl wird mit VG gekennzeichnet. Durch Klicken mit der rechten Maustaste wählen Sie eine Hintergrundfarbe aus. Diese Auswahl wird mit HG gekennzeichnet. Wird die Farbe nach dieser Aktion mit VH gekennzeichnet, wird die Farbe sowohl für den Hintergrund als auch für den Vordergrund verwendet.
4. Wählen Sie gegebenenfalls ein Attribut, wie z.B. Fettschrift, aus.
5. Wenn Sie für den Vordergrund (VG) oder den Hintergrund (HG) die Standardfarben wählen, werden diese für das Element verwendet.
6. Wiederholen Sie die Schritte 2 bis 4 für alle Elemente, die Sie modifizieren wollen.

Meldungsfenster

Durch Auswahl von *Vorgaben* im Dialogfenster *IDE-Einstellungen* können Sie die Funktionalität der Meldungsfenster steuern.

Im Meldungsfenster werden während der Programmcompilierung Fehlermeldungen und Warnungen angezeigt. Wenn Sie im Meldungsfenster eine Meldung auswählen, setzt der Editor den Cursor an die Stelle Ihres Programms, auf die sich die Fehlermeldung oder Warnung bezieht. Ist die Datei, in der der Fehler auftrat, in keinem Editorfenster geladen, können Sie sie durch Drücken der Leertaste laden. Die Datei kann auch mit der Tastenkombination [*Alt*]+[*F10*] und mit *Quelltext öffnen* im lokalen Menü geladen werden. Das Meldungsfenster bleibt geöffnet, so daß Sie die Meldungen nacheinander bearbeiten können.

Wenn Sie sich den Programmteil anzeigen lassen möchten, auf den sich die Fehlermeldung oder die Warnung bezieht, wählen Sie entweder die Meldung im Meldungsfenster aus und drücken die Eingabetaste oder doppelklicken auf die Meldung oder drücken die Tastenkombination [*Alt*]+[*F10*] und wählen im lokalen Menü *Quelltext öffnen* aus. Der Cursor wird an die Position des Quelltextes bewegt, an der der Fehler verursacht

wurde. Das Meldungsfenster wird in den Hintergrund verschoben. Mit der Tastenkombination [*Alt*]+[*F7*] springen Sie zur nächsten, mit der Tastenkombination [*Alt*]+[*F8*] zur vorherigen Fehlermeldung.

Sie haben auch die Möglichkeit, mit dem Cursor in der Meldungsliste zu blättern. Wenn Sie eine Meldung auswählen, wird der Cursor im Editorfenster an die Stelle bewegt, an der der Fehler auftrat. Diese Funktion wird als automatische Fehlerrückverfolgung bezeichnet. Die automatische Fehlerrückverfolgung ist nur aktiviert, wenn die Datei, die den Fehler enthält, in einem Editorfenster angezeigt wird. Verweist die nächste ausgewählte Fehlermeldung auf eine andere Quelldatei (also nicht auf die Datei des aktuellen Editorfensters), müssen Sie das Editorfenster aktivieren, das die entsprechende Quelldatei enthält, bevor Sie die automatische Fehlerrückverfolgung weiterführen können.

Sie können das Meldungsfenster löschen, indem Sie im lokalen Menü des Meldungsfensters *Alle Meldungen entfernen* auswählen. Zum Aufruf des lokalen Menüs klicken Sie auf die rechte Maustaste oder drücken die Tastenkombination [*Alt*]+[*F10*].

Programm durchsehen

Mit der Symbolanzeige können Sie sich Objekthierarchien, Klassen, Funktionen, Variablen, Typen und Konstanten anzeigen lassen, die Ihr Programm verwendet. Außerdem können Sie

- die Hierarchien Ihrer Anwendung grafisch darstellen, ein beliebiges Objekt auswählen und die Funktionen und Symbole, die es enthält und erbt, anzeigen lassen;
- die Variablen auflisten, die Ihr Programm definiert, und die Deklaration einer beliebigen Variablen anzeigen, alle Verweise in Ihrem Programm auf diese Variable auflisten oder die Deklaration in Ihrem Quelltext editieren.

Bevor sie mit der Symbolanzeige arbeiten, muß sichergestellt sein, daß die folgenden Optionen im Dialogfenster *Projekt-Optionen* (wählen Sie zuvor *Optionen/Projekt* aus) gesetzt wurden und Ihr Programm compiliert wurde.

- Wählen Sie *Compiler/Debugger* aus, und markieren Sie *Debug-Information in .OBJ-Dateien*.
- Wählen Sie *Compiler/Debugger* aus, und markieren Sie *Informationen für Symbolanzeige in .OBJ-Dateien*.
- Wählen Sie *Linker/Allgemein* aus, und markieren Sie *Debug-Informationen mit aufnehmen*.

Die Symbolanzeige verfügt über eine konfigurierbare Mauspalette.

Wählen Sie zum Aufrufen der Symbolanzeige *Suchen/Symbol anzeigen*, *Anzeige/Klassen* oder *Anzeige/Globale Variablen* aus. Sie können die Symbolanzeige auch aufrufen, indem Sie den Cursor auf ein Symbol Ihres Programms bewegen und *Suchen/Symbol anzeigen* auswählen. Wenn das Programm im aktuellen Editorfenster noch nicht compiliert wurde, müssen Sie es jetzt mit Debug-Informationen compilieren und linken, da Sie sonst die Symbolanzeige nicht verwenden können. Wenn Sie versuchen, eine Klassendefinition oder ein Symbol, für das keine Debug-Daten vorhanden sind) durchzusehen, wird eine Fehlermeldung ausgegeben.

Im Dialogfenster *IDE-Einstellungen* können Sie verschiedene Optionen für die Symbolanzeige setzen. Wählen Sie dazu *Optionen/IDE* aus, klicken Sie *Symbolanzeige* an und wählen Sie die Optionen aus, die Sie verwenden wollen. *Ein Fenster* bedeutet, daß Sie jeweils nur ein Symbolanzeigefenster geöffnet haben können. *Mehrere Fenster* bedeutet, daß bei jeder neuen Aktion, die Sie über die Symbolanzeige durchführen, wie z.B. der Auswahl *Anzeige/Globale Variablen* im Hauptmenü, ein neues Symbolanzeigefenster geöffnet wird.

Objekte durchsehen (Klassenübersicht)

Durch Auswählen von *Anzeige/Klassen* können Sie sich sowohl einen groben Überblick über die Objekthierarchien in Ihrem Programm verschaffen als auch Einzelheiten anzeigen lassen. Nach Auswahl dieser Option zeichnet die Symbolanzeige die Objekte und stellt ihre Vorfahre-Nachfahre-Beziehungen mit einer Baumstruktur dar. Die roten Linien sollen helfen, die direkten Vorfahre-Nachfahre-Beziehungen des aktuell ausgewählten Objekts zu erkennen. Abbildung 1.2 zeigt die Struktur des Programms WHELLO.

Abbildung 1.2: Klassen in einem Programm anzeigen



Möchten Sie sich weitere Informationen über ein bestimmtes Objekt anzeigen lassen, doppelklicken Sie auf das Objekt. Wenn Sie nicht mit der Maus arbeiten, wählen Sie das Objekt mit den Pfeiltasten aus und drücken die Eingabetaste.

Sie können auch die Symbolanzeige-Optionen im Dialogfenster *IDE-Einstellungen* markieren, um einen Symboltyp auszuwählen. Dabei müssen Sie die Optionen jedoch setzen, bevor Sie das Symbolanzeigefenster öffnen.

Beim Anzeigen eines bestimmten Symbols werden die Buchstaben, die das Symbol bezeichnen, in einer Filtermatrix am unteren Rand des Symbolanzeigefensters angezeigt. Mit Filtern können Sie die Symboltypen auswählen, die angezeigt werden sollen. (Tabelle 1.1 enthält die Liste der Buchstaben und deren Bedeutungen.)

Die Filtermatrix enthält für jeden Buchstaben eine Spalte. Um einen Buchstaben zu verschieben, klicken Sie die obere oder untere Zeile an. Ein Buchstabe in der oberen Zeile bedeutet, daß Symbole mit dieser Bezeichnung angezeigt werden; ein Buchstabe in der unteren Zeile bedeutet, daß Symbole mit dieser Bezeichnung nicht angezeigt werden.

Wenn ein bestimmter Symboltyp nicht angezeigt werden soll, klicken Sie die untere Zelle der Spalte des Buchstabens an (siehe Abbildung links). Möchten Sie beispielsweise alle im aktuell ausgewählten Objekt angezeigten Variablen entfernen, klicken Sie die untere Zelle in der Spalte **V** an.



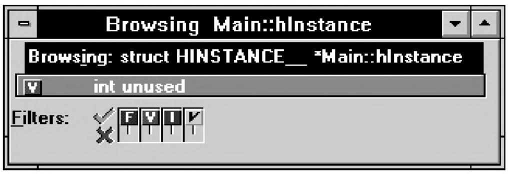
In manchen Fällen werden neben einem Symbol mehrere Buchstaben angezeigt. Der zweite Buchstabe erscheint direkt hinter dem Buchstaben, der den Symboltyp bezeichnet, und beschreibt das Symbol näher. Tabelle 1.1 enthält eine Liste der Filterbezeichner.

Deklarationen aufgelisteter Symbole anzeigen

Sie können die Deklaration eines bestimmten, in einer Liste angezeigten Symbols wie folgt anzeigen:

- Doppelklicken Sie auf das Symbol.
- Wählen Sie das Symbol aus und drücken Sie die Eingabetaste.
- Wählen Sie das Symbol aus, rufen Sie durch Drücken der Tastenkombination [*Alt*]+[*F10]* das lokale Menü auf, und wählen Sie *Symbol anzeigen* aus. Die Symboldeklaration erscheint, wie in Abbildung 1.3 gezeigt, in einem Fenster.

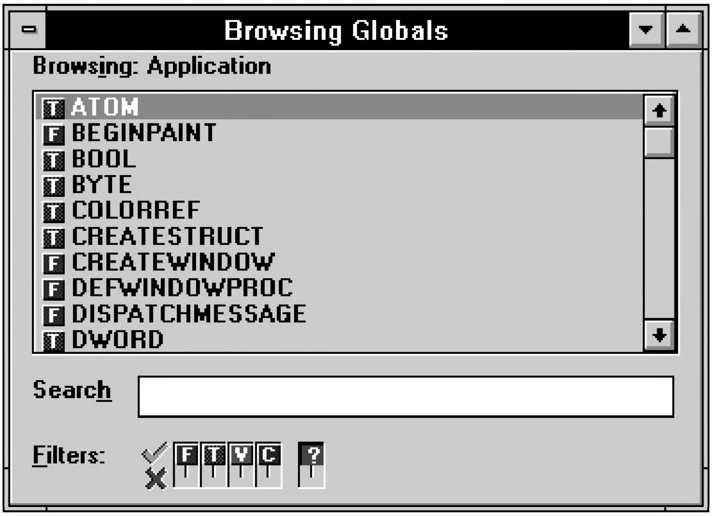
Abbildung 1.3: Fenster für Symboldeklaration



Globale Symbole anzeigen

Wenn Sie *Anzeige/Globale Symbole* auswählen, wird ein Fenster angezeigt, in dem alle globalen Symbole Ihres Programms in alphabetischer Reihenfolge aufgeführt sind. Die Symbolanzeige listet die im Objekt verwendeten Symbole (Funktionen, Variablen usw.) auf. Abbildung 1.4 zeigt die globalen Symbole für das Programm WHELLO.

Abbildung 1.4: Anzeige von globalen Symbolen



Links neben den einzelnen Symbolen im Objekt werden ein oder mehrere Buchstaben angezeigt. Die Buchstaben beschreiben die Art des Symbols. Über die Filterliste am unteren Rand des Symbolanzeigefensters können Symbole ausgefiltert werden. Weitere Informationen dazu finden Sie im vorhergehenden Text.

Tabelle 1.1: Buchstabensymbole in der Symbolanzeige

Buchstabe	Symbol
F	Funktionen
T	Typen
V	Variablen
C	Ganzzahlige Konstanten
?	Debugfähig
I	Geerbte
v	Virtuelle Methode

Sie können zum Suchen auch reguläre Ausdrücke, wie z.B. ?, * und + verwenden.

Wenn Sie weitere Informationen zu einem Symbol aufrufen möchten, klicken Sie das Symbol an oder wählen es mit den Pfeiltasten aus. In einem Eingabefeld am unteren Rand des Fensters können Sie die Liste der globalen Symbole schnell durchsuchen. Geben Sie dazu die Anfangsbuchstaben des Symbolnamens ein. Während der Eingabe wird der Auswahlbalken auf ein Symbol bewegt, das mit den eingegebenen Buchstaben übereinstimmt. Zum Anzeigen der Deklaration des Symbols wählen Sie dieses aus und drücken die Eingabetaste. Weitere Informationen dazu finden Sie im vorherigen Abschnitt »Deklarationen aufgelisteter Symbole anzeigen«.

Mit regulären Ausdrücken in der Symbolanzeige suchen

In einigen Symbolanzeigefenstern können Sie Ausdrücke im Suchfenster verwenden. Tabelle 1.2 enthält eine Liste der zulässigen Symbole.

Tabelle 1.2: Suchausdrücke der Symbolanzeige

Zeichen	Funktion
.	Sucht ein übereinstimmendes Zeichen.
*	Sucht keine oder mehrere Übereinstimmungen für das vorausgehende Zeichen. Beispiele: * ist falsch, da kein Zeichen vorausgeht; fo* sucht alle Symbole, die mit dem Buchstaben "f" beginnen; fo*x stimmt mit "fx", "fox", "fooox" überein.
+	Sucht eine oder mehrere Übereinstimmungen für das vorausgehende Zeichen. Beispiele: + ist falsch; fo+ sucht alle Symbole, die mit "fo" beginnen; fo+x stimmt mit "fox", "fooox" überein.
?	Sucht keine oder eine Übereinstimmung mit den vorausgehenden Zeichen. Beispiele: ? ist falsch; fo? sucht alle Symbole, die mit dem Buchstaben "f" beginnen; fo?x stimmt nur mit "fx" oder "fox" überein.

Symbole im eigenen Programm anzeigen

Sie können auch jedes Symbol Ihres eigenen Programms anzeigen, ohne die Objekthierarchien oder die Symbollisten vorher aufzurufen. Dabei haben Sie zwei Möglichkeiten:

- Heben Sie das Symbol in Ihrem Programm hervor, und wählen Sie *Suchen/Symbol anzeigen* aus.
- Rufen Sie in einem Editorfenster das lokale Menü durch Klicken auf die rechte Maustaste oder Drücken der Tastenkombination [*Alt*]+[*F10*] auf, und wählen Sie *Symbol anzeigen* aus.

K A P I T E L 2

Erstellen und Verwalten von Projekten

Die integrierte Entwicklungsumgebung (IDE) von Borland C++ enthält eine Projektverwaltung, die eine grafische Darstellung der in Ihrem Projekt enthaltenen Dateien liefert. Mit dem Projekt-Manager können Sie genau sehen, welche Dateien Sie erzeugen, welche Dateien Sie dabei verwenden und welche Optionen Sie gesetzt haben.

Dieses Kapitel umfaßt folgende Themen, die die Verwendung des Projekt-Managers zur Organisation der Dateien in Ihrem Projekt beschreiben:

- Projektverwaltung
- Verwendung des Projekt-Managers
- Gruppieren von Dateisätzen mit Source Pools
- Übersetzungs- und Hilfsprogramme und Viewer

Informationen zum Setzen von Projektoptionen und zur Compilierung finden Sie in Kapitel 3.

Was ist Projektverwaltung?

Wenn Ihre Anwendungen an Größe und Komplexität zunehmen, werden Sie feststellen, daß Ihre Programme immer mehr von verschiedenen Zwischendateien abhängig werden. Sie werden außerdem bemerken, daß einige Module in Ihrem Projekt mit verschiedenen Compilern oder unterschiedlichen Compiler-Optionen compiliert werden müssen. Ein Windows-Programm kann sich beispielsweise aus Ressourcenskripts, Importbibliotheken und Quell-Code zusammensetzen, wobei jeder Dateityp einen anderen Compiler und eigene Compiler-Einstellungen erfordert.

Wenn die Komplexität Ihrer Projekte zunimmt, wächst der Bedarf an einer Methode zur Verwaltung der verschiedenen Komponenten Ihrer Projekte. Wenn Sie sich die zu einem Projekt zusammengefaßten Dateien ansehen, können Sie feststellen, wie ein Projekt verschiedene *Quelldateien* kombiniert, um verschiedene *Zieldateien* zu erzeugen. Zieldateien können z.B. .DLL- oder .EXE-Dateien sein. Die Quelldateien, von denen diese abhängen, können aus Dateien wie .C-, .CPP- oder .H-Dateien bestehen. *Projektverwaltung* ist die Organisation und Verwaltung der Quell- und Zieldateien, die Ihr Projekt bilden.

Verwendung des Projekt-Managers

Der Projekt-Manager stellt die Dateien Ihres Projekts als Hierarchiediagramm in einem *Projektbaum* dar. Jede Hierarchieebene enthält ein einzelnes Ziel mit den davon abhängigen Dateien. Die Abhängigkeitsverhältnisse werden durch Einrückungen dargestellt, wie in Abbildung 2.1 gezeigt. Verwenden Sie die Symbole + und -, um Hierarchieebenen anzuzeigen oder zu verbergen.

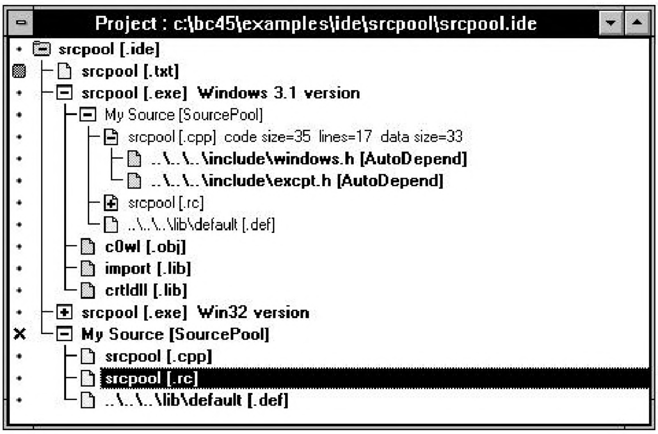


Abbildung 2.1: Darstellung eines Projektbaums im Projektfenster

Der Projektmanager verwendet folgende Arten von *Knoten*, um die verschiedenen Dateitypen eines Projekts voneinander zu unterscheiden:

- Der an der Spitze des Projektbaums befindliche *Projektknoten* steht für das gesamte Projekt. Jede zum Aufbau des Projekts notwendige Datei erscheint unter diesem Projektknoten (einen Projektknoten kann man sich als symbolisches Ziel einer Make-Datei vorstellen). Per Voreinstellung erscheint der Projektknoten nicht im Projektbaum.
 Wählen Sie, um ihn anzuzeigen, *Optionen/Umgebung*, dann *Projektansicht* aus der Themenliste und klicken Sie auf *Projektknoten anzeigen*.

- Ein *Zielknoten* steht für eine Datei, die erzeugt wird, wenn ihre abhängigen Knoten aufgebaut werden (ein Zielknoten ist üblicherweise eine aus Ihrem Quell-Code erzeugte .EXE-, .DLL- oder .LIB-Datei). Ein Projekt kann zahlreiche Zielknoten enthalten. In einem einzelnen Projekt könnten Sie beispielsweise eine ausführbare Datei und zwei eigene .DLL-Dateien erzeugen, dies ergibt insgesamt drei Zieldateien.
- *Quellknoten* stehen für Dateien, die zum Aufbau eines Ziels verwendet werden. Dateien wie .C, .CPP und .RC sind solche Quellknoten.
- Ein *Laufzeitknoten* bezeichnet Dateien, die der Projekt-Manager während des Linkens Ihres Projekts verwendet, wie Startup-Code und .LIB-Dateien. Abhängig von den in TargetExpert festgelegten Optionen fügt der Projekt-Manager verschiedene Laufzeitknoten hinzu. Per Voreinstellung erscheinen Laufzeitknoten nicht im Projekt-Manager. Wählen Sie, um sie anzuzeigen, *Optionen/Umgebung/Projektansicht* und markieren Sie *Laufzeitknoten anzeigen*.
- *Autodependency-Knoten* sind Dateien, auf die Ihr Programm automatisch zugreift, wie z.B. eingebundene Header-Dateien. Wenn Sie Autodependency-Knoten anzeigen, können Sie die von Quellknoten abhängigen Dateien sehen und vom Projekt-Manager aus leicht auf sie zugreifen. Autodependency-Knoten erscheinen standardmäßig nicht im Projekt-Manager. Um sie anzuzeigen, wählen Sie *Optionen/Projekt/Erstellen* und markieren Sie *Autodependencies: Verbergen & Anzeigen*. Beachten Sie, daß ein Projekt erstellt sein muß, bevor der Projekt-Manager Autodependency-Informationen anzeigen kann.

Für die Knoten verwendet der Projekt-Manager folgendes Farbschema:

- Blaue Knoten sind vom Programmierer hinzugefügt.
- Weiße Knoten bezeichnen Projektzieldateien.
- Gelbe Knoten werden vom Compiler hinzugefügt (wenn er Dependencies und Autodependencies einträgt), von AppExpert oder ClassExpert (wenn sie .CPP-Knoten hinzufügen) oder von TargetExpert (wenn er auf dem Zieltyp basierende Knoten hinzufügt).

Der Projekt-Manager verwendet spezielle *Zeichenkörper* am linken Rand, um die Aufbauattribute von Projektknoten anzuzeigen. Um an einem Knoten Aufbauattribute anzuwenden (und um die verschiedenen Arten von Zeichenkörpern kennenzulernen), wählen Sie zuerst *Lokale Optionen* aus dem lokalen Menü des Projekt-Managers und dann *Aufbauattribute*.

Zusätzlich können Sie zur Unterstützung bei der Organisation Ihrer Projektdateien den Projekt-Manager dazu verwenden, auf Quelldateien zuzugreifen und Zieldateien zu erstellen.

- Um eine Quelldatei in ein Editierfenster zu laden, doppelklicken Sie auf den Knoten im Projektbaum oder markieren Sie ihn und drücken Sie

entweder [*Eingabe*] oder wählen Sie *Ansicht/Texteditor* aus dem lokalen Menü des Projekt-Managers.

- Es ist sehr effektiv, den Projekt-Manager zum *Erstellen* eines Projekts zu verwenden, da sie ihn veranlassen können, nur diejenigen Dateien zu *übersetzen*, die sich seit dem letzten Projektaufbau verändert haben; so werden keine Computer-Ressourcen für unnötige Aktualisierungen von Dateien verschwendet. (Der Ausdruck »übersetzen« bezeichnet die Verwendung eines Dateityps zur Erzeugung eines anderen. Der C++ Compiler etwa ist ein Übersetzer, da er .OBJ-Dateien aus .CPP-Dateien generiert.)

Es gibt verschiedene Methoden zur Anpassung der Aufbauoptionen der Knoten Ihres Projekts. Der folgende Abschnitt »Festlegung von Projektoptionen und Compilierung« enthält eine ausführliche Beschreibung der Pflege von Projektoptionen und der Compilierung von Projekt-Zieldateien.

Anlegen eines Projekts

Wenn Sie mit dem Schreiben eines neuen Programms beginnen, ist der erste Schritt das Anlegen eines neuen Projekts zur Organisation Ihrer Programmdateien. Der Befehl *Projekt/Neues Projekt* öffnet das Dialogfenster *Neues Ziel*.

Setzen von Optionen für Zieldateien mit dem Dialogfenster *Neues Ziel*

Wenn Sie ein neues Projekt anlegen, weist die IDE den Knoten Ihres Projekts automatisch Standard-Dateinamen zu. Die folgenden Schritte zeigen, wie Sie diese Voreinstellungen ändern können und wie die Anfangs-Projekteinstellung vervollständigt wird.

1. Geben Sie Pfad und Namen für das neue Projekt in das Eingabefeld *Projektpfad und -name* ein (ein Projektname besteht aus acht oder weniger Zeichen). Die Eingabe einer Dateinamenerweiterung ist nicht notwendig, da die IDE allen Projektdateien automatisch die Erweiterung .IDE zuweist.
2. Geben Sie den Namen der ersten Zieldatei Ihres Projekts in das Eingabefeld *Zieldateiname* ein. Dies ist normalerweise der Name der .EXE- oder .DLL-Datei, die Sie erzeugen wollen.

Hinweis

Die übrigen Felder im Dialogfeld *Neues Ziel* setzen Optionen für das erste Ziel im Projekt. Sie werden gewöhnlich gesammelt als *TargetExpert* bezeichnet, da sie Bestandteile des Dialogfensters *TargetExpert* sind (siehe Abbildung 2.2)

Abbildung 2.2:
Das Dialogfenster
TargetExpert



3. Wählen Sie den gewünschten Zieldateityp aus der Zieldateitypen-Liste. Informationen zu den Dateitypen finden sie in der Online-Hilfe zur IDE. Informationen zur Verwendung des Hilfe-Compilers finden Sie in der Online-Hilfedatei CWH.HLP.
4. Wählen Sie eine Zielumgebung aus der Dropdown-Liste *Umgebung*. Informationen zu den einzelnen Umgebungstypen finden Sie in der Online-Hilfe.
5. Wählen Sie das Speichermodell für die Zieldatei aus den Zielmodell-Optionen.
 - **Tiny**, das für reine 16-Bit-DOS-Anwendungen zur Verfügung steht, setzt vier Segmentregister (CS, DS, SS und ES) auf dieselbe Startadresse. Sie erhalten 64 KB Speicher für Code, Daten und Stacks.
 - **Small** verwendet unterschiedliche Code- und Datensegmente. Sie erhalten near-Code und near-Daten.
 - **Medium** liefert near-Daten und far-Code.
 - **Compact** ist die Umkehrung des Medium-Speichermodells. Sie erhalten near-Code und far-Daten.
 - **Large** liefert far-Code und far-Daten.
 - **Huge** wird mit reinen 16-Bit-DOS-Anwendungen verwendet und gleicht dem Large-Modell, läßt jedoch mehr als 64 KB statischer Daten zu.
 - **GUI** wird für Win32 GUI-Anwendungen verwendet.
 - **Console** wird mit Win32 Textbildschirmanwendungen verwendet.

Hinweis

Windows-und DOS-Programme verwenden DS- und SS-Segmente unterschiedlich. In Windows sind es stets dieselben, in DOS dagegen sind diese Segmente für Compact-, Large- und Huge-Speichermodel-

le verschieden. Weitere Informationen zu Speichermodellen finden Sie im *DOS Referenzhandbuch*.

Die Einträge in der Zielmodelliste stellen sich gemäß dem gewählten Umgebungstyp ein.

6. Wählen Sie unter Verwendung der Gruppe *Standardbibliotheken* die Typen von Bibliotheken, die mit Ihrer Anwendung verwendet werden sollen (abhängig von der Umgebung und den Zielauswahlkriterien werden einige dieser Bibliotheken eventuell nicht angezeigt):

 - **OWL** verwendet die ObjectWindows-Bibliotheken. Weitere Informationen finden Sie im *ObjectWindows Programmierhandbuch*.
 - **OCF** verwendet die Hilfebibliothek OCOLE.DLL (ein Gerüst, das OLE 2 kapselt) von ObjectComponents. Weitere Informationen finden Sie im *ObjectWindows Programmierhandbuch*.
 - **Klassenbibliothek** verwendet die im *Programmierhandbuch* und im *Referenzhandbuch* erläuterten Borland C++ Container-Klassen-Bibliotheken.
 - **Laufzeitbibliothek** verwendet die im *Referenzhandbuch* aufgelisteten C/C++ Standardfunktions-Bibliotheken.
 - **BWCC** benützt die Bibliotheken der Dialogelemente im Borland-Stil. Näheres hierzu in Kapitel 14.
 - **OLE 2** linkt in OLE 2 unterstützenden Bibliotheken.
 - **BGI**, das für reine DOS-Anwendungen zur Verfügung steht, verwendet die Borland Graphics Interface-Bibliotheken. Weitere Informationen zum BGI finden Sie im *DOS Referenzhandbuch*.

 Für Windows-Anwendungen können Sie unter folgenden Typen von Link-Methoden wählen:

 - **Dynamic** bewirkt, daß Ihre Anwendung die Standardbibliotheken während der Laufzeit einbindet – die Funktionen in den Standardbibliotheken werden, anstatt ihrer Anwendung direkt hinzugefügt zu werden, in einer separaten .DLL untergebracht. Damit verringert sich die Größe eines Programms beträchtlich, doch ist das Ziel in diesem Falle vom Vorhandensein der .DLLs während der Laufzeit abhängig.
 - **Static** bindet die Standard-Bibliotheksfunktionen direkt in Ihre ausführbare Datei ein und erzeugt so ein größeres, eigenständiges ausführbares Programm. Wenn Sie viele .DLLs mit statisch eingebundenen RTLs erzeugen, erhält jede .DLL eine eigene Kopie der von ihr benutzten Routinen.

 Ist Ihre Anwendung für DOS, wählen Sie eine der folgenden Bibliotheken zur Unterstützung des verwendeten Gleitkommatyps:

- **Floating point** nutzt mathematische Koprozessoren und bindet FP87.LIB ein.
- **Emulation** bindet keine Arithmetikbibliotheken; Hardware-Unterstützung wird emuliert.
- **No Math Support** für Programme ohne Gleitkomma-Arithmetik.

Die folgenden auswählbaren Bibliotheken beziehen sich auf sehr spezifische Situationen der Programmerzeugung:

- **Alternate Startup** (nur für DOS-Programme) bindet C0Fx.OBJ ein, das für alle Speichermodelle SS==DS setzt.
- **Diagnostic** verwendet eine Diagnoseversion der Bibliotheken, die den Dateien Debugging- und Diagnoseinformationen hinzufügen. Diese Option steht für reine Klassenbibliotheken und ObjectWindows zur Verfügung.

Hinweis Wenn Sie die CD-Version von Borland C++ besitzen, stehen die Diagnoseversionen der Bibliotheken auf der CD zur Verfügung. Haben Sie die Diskettenversion von Borland C++, so müssen Sie die Diagnoseversionen der Bibliotheken unter Verwendung der Kommandozeilenwerkzeuge erzeugen. Informationen über die zur Erstellung dieser Bibliotheken benötigten Kommandozeilenoptionen finden Sie in den Kommentaren zu Beginn der in folgenden Verzeichnissen befindlichen MAKEFILE-Dateien:

\BC45\SOURCE\CLASSLIB
(für die Kommandozeilen der Klassenbibliotheken)

\BC45\SOURCE\OWL
(für die Kommandozeilen der OWL Bibliotheken)

- **Multithread** benutzt die Mehrfachpfadversion der Laufzeitbibliothek. Verwenden Sie diese Option, wenn Sie Mehrfachpfadanwendungen für Win32-Betriebssysteme (wie z.B. Windows NT) erstellen.

Hinweis Manche Bibliotheken werden bei bestimmten Zieltypen automatisch markiert (einige davon können Sie nicht deaktivieren, da sie für den gewählten Zieltyp benötigt werden). Gibt es statische und dynamische Bibliotheken, so können Sie den gewünschten Bibliothekstyp bestimmen (meist wohl *Dynamic*).

7. Wenn notwendig, klicken Sie auf die Schaltfläche *Erweitert*, um die mit dem neuen Ziel erzeugten Typen von Knoten näher zu bestimmen (im folgenden Abschnitt finden Sie eine Beschreibung hierzu).
8. Klicken Sie auf OK, um die Einstellungen zu bestätigen und das Dialogfenster *Neues Ziel* zu schließen. Der Projekt-Manager erzeugt die Projektdatei und versieht sie mit der Erweiterung .IDE.

Wenn sie das Dialogfenster *Neues Ziel* schließen, stellt der Projekt-Manager das Projekt im Projektfenster grafisch dar. Über den Projektknoten erzeugt der Projekt-Manager einen Zielknoten mit einem oder mehreren bestimmenden Knoten. Nach dem Erstellen eines Anfangsziels für einen Knoten können Sie ihrem Projekt Knoten hinzufügen oder die vorhandenen löschen oder verändern, wie im folgenden erläutert.

Festlegung des Typs bestimmender Knoten

Die Schaltfläche *Erweitert* im Dialogfenster *Neues Ziel* öffnet das Dialogfenster *Erweiterte Optionen*. Verwenden Sie dieses Dialogfenster, um die Typen von bestimmenden Knoten festzulegen, die die IDE zusammen mit einem neuen Zielknoten erzeugt.

- **.CPP-Knoten** erzeugt einen bestimmenden Knoten in C++ Form.
- **.C-Knoten** erzeugt einen bestimmenden Knoten in Form der Sprache C.
- **Kein Knoten** erzeugt einen Zielknoten, der keinen bestimmenden Knoten verwendet. Benutzen Sie diese Option, wenn Sie einen bestimmenden Knoten erzeugen möchten, der denselben Dateinamen wie das Projekt trägt. Wenn Sie mit dieser Option einen neuen Zielknoten erzeugen, müssen Sie den bestimmenden Knoten ausdrücklich hinzufügen, wie unter »Hinzufügen von bestimmenden Knoten zu einem Projekt« beschrieben.

Ist Ihr Programm für Windows geschrieben, markieren Sie folgende Felder entsprechend den im Ziel verwendeten Dateien:

- **.RC** erzeugt einen bestimmenden Knoten, der mit einem Ressourcenskript verbunden wird.
- **.DEF** erzeugt einen bestimmenden Knoten, der mit einer Windows-Definitionsdatei verbunden wird.

Öffnen bestehender Projekte

Wählen Sie, um ein bestehendes Projekt zu öffnen, *Projekt/Projekt Öffnen* und verwenden Sie dann die Datei-Symbolanzeige, um eine bestehende .IDE- oder .PRJ-Projektdatei auszuwählen (beim Abspeichern des Projekts werden .PRJ- in .IDE-Dateien umgewandelt). Wird das Projekt geöffnet, ohne daß das Projektfenster sichtbar ist, so können Sie dorthin über *Ansicht/Projekt* gelangen.

Hinzufügen von bestimmenden Knoten zu einem Objekt

Um einem Objekt bestimmende Knoten hinzuzufügen, führen Sie folgende Schritte aus:

1. Wählen Sie irgendeinen Knoten im Projektbaum, unter dem der neue Knoten erscheinen soll. Soll der Knoten beispielsweise unter dem Ziel erscheinen, wählen Sie den Zielknoten.

2. Drücken Sie [*Einfg*], klicken Sie auf den Schalter in der Symbolleiste oder klicken Sie mit der rechten Maustaste auf den Knoten, um das lokale Menü des Projektfensters zu öffnen. Wählen Sie dann *Knoten hinzufügen*.
3. Verwenden Sie die Dateisymbolanzeige, um die Datei oder die Dateien auszuwählen, die mit dem neuen Knoten verbunden werden sollen. Sie können auch den Namen der hinzuzufügenden Datei eingeben.
4. Wählen Sie OK, um die Einstellungen zu bestätigen.

Sie können den Windows-Datei-Manager verwenden, um einen oder mehrere bestimmende Knoten hinzuzufügen.

1. Öffnen Sie den Datei-Manager und orden Sie die Fenster so an, daß das Projektfenster in der IDE noch sichtbar ist.
2. Drücken Sie [*Strg*] im Datei-Manager und wählen Sie die Dateien aus, die als bestimmende Knoten hinzugefügt werden sollen.
3. Ziehen Sie die Dateien vom Datei-Manager auf einen Knoten im Projektfenster. Der Projekt-Manager fügt die bestimmenden Knoten automatisch unter dem ausgewählten Knoten hinzu.

Löschen von bestimmenden Knoten

Um einen Knoten in einem Projekt zu löschen, markieren Sie diesen und drücken Sie [*Entf*] oder wählen Sie *Knoten löschen* aus dem lokalen Menü. Um mehrere Knoten zu löschen, markieren Sie die zu löschenden Knoten (drücken Sie [*Strg*] oder [*Umschalt*] und klicken Sie mit der linken Maustaste, um mehrere Knoten zu markieren) und drücken Sie [*Entf*]. Der Projekt-Manager fragt, ob Sie die Knoten löschen wollen, bevor er fortfährt.

Bearbeiten der Eigenschaften bestimmender Knoten

Knoten-Attribute beschreiben die Eigenschaften von Knoten und definieren (sofern anwendbar) das Übersetzungswerkzeug. Die Eigenschaften bestimmender Knoten werden wie folgt geändert:

1. Klicken Sie mit der rechten Maustaste den bestimmenden Knoten an (oder markieren Sie den Knoten und drücken Sie [*Alt*][*F10*]). Wählen Sie im lokalen Menü *Knotenattribute bearbeiten*. Es erscheint das Dialogfenster *Knotenattribute*.
2. Verändern Sie die Eigenschaften nach Wunsch und bestätigen Sie diese mit OK.

 Knotenattribute sind folgendermaßen definiert:

 - **Name** ist die Bezeichnung des Knotens ohne Dateinamenerweiterung.
 - **Beschreibung** ist ein beliebiger erläuternder Text für den Knoten.
 - **Schablone** ist der Name eines Optionssatzes, den der Projekt-Manager bei der Übersetzung des Knotens verwendet. Wird *Keine* ange-

geben, verwendet der Projekt-Manager die Optionseinstellungen der Schablone des übergeordneten Fensters. Weitere Informationen zu Optionssätzen finden Sie im nächsten Kapitel, »Festlegung von Objekteigenschaften und Compilierung«.

- **Übersetzer** benennt das auf diesen Knoten angewendete transformierende Programm. Die IDE weist dem Knotentyp einen Vorgabeübersetzer zu (z.B. CppCompile für einen .CPP-Knoten), der in diesem Feld überschrieben werden kann. Weitere Informationen finden Sie unter »Übersetzer, Viewer und Dienstprogramme«.
- **Knotentyp** definiert die Namenserweiterung für den Knoten, die ihrerseits die für den Knoten verfügbaren Übersetzer festlegt.

Hinzufügen von Zielknoten zu einem Projekt

Um einem Projekt mit dem Dialogfenster *Neues Ziel* ein Ziel hinzuzufügen,

1. Wählen Sie *Projekt/Neues Ziel* oder klicken Sie auf die Symbolleisten-Schaltfläche.
2. Geben Sie den Namen für das neue Ziel ein und wählen Sie einen der folgenden Zieltypen:
 - **Standard** (Vorgabe) kann eine ausführbare, .DLL- oder andere Datei sein.
 - **AppExpert** ist eine automatisch erzeugte, auf ObjectWindows basierende Anwendung. In Kapitel 4 finden Sie weitere Informationen zu diesem Typ.
 - **Source Pool** ist eine Sammlung von Dateien , auf die in anderen Zielen Bezug genommen werden kann. Sie finden nähere Informationen zur Verwendung von Quelltextsammlungen weiter unten.
3. Wählen Sie OK. Ist der Zieltyp *Standard*, erscheint das Dialogfenster *TargetExpert*, in dem Sie das neue Ziel näher definieren können. Ist der Zieltyp *Source Pool*, so wird das Ziel ins Projekt eingefügt und Sie können ihm sofort Knoten hinzufügen. Wählen Sie *AppExpert* als Ziel, erscheint AppExpert.

Wenn Sie ein neues Ziel hinzufügen, wird dieses stets am Ende des Projektbaums angefügt.

Ein Beispiel für ein Projekt mit zwei Zieldateien ist MULTITRG.IDE im Verzeichnis \BC45\EXAMPLES\IDE\MULTITRG. Diese Projektdatei erzeugt eine 16-Bit- und eine 32-Bit-Version des Programms WHELLO. Das Projekt enthält eine Textdatei, die beschreibt, wie ein oder mehrere Ziele in einem Projekt verwendet werden.

Hat ein Projekt mehr als ein Ziel, so können Sie wählen, ob Sie ein einzelnes Ziel, mehrere Ziele oder das gesamte Projekt erstellen möchten. Nähere Informationen finden Sie unter »Projekte compilieren«.

Zielknoten löschen

Um einen Zielknoten zu löschen:

1. Klicken Sie mit der rechten Maustaste auf den zu löschenden Zielknoten (oder markieren Sie ihn und drücken Sie [*Alt*][*F10*]).
2. Wählen Sie *Knoten löschen* aus dem lokalen Menü.
3. Der Projekt-Manager fragt, ob Sie das Ziel ganz sicher löschen wollen. Wählen Sie OK, um das Ziel und alle seine Untereinheiten aus dem Projekt zu entfernen.

Sie können mehrere Knoten auch löschen, indem Sie [*Strg*] drücken und auf die zu löschenden Knoten klicken. Drücken Sie dann [*Entf*].

Achtung

Vorsicht beim Löschen von Zielknoten – Sie können diesen Vorgang nicht rückgängig machen.

Bearbeiten von Zieleigenschaften mit TargetExpert

Zieleigenschaften beschreiben den Typ eines Ziels. Ein Ziel kann beispielsweise eine 32-Bit-Windows-DLL oder eine ausführbare 16-Bit-DOS-Datei sein. Unter Verwendung von TargetExpert (siehe Abbildung 2.2) können Sie die Eigenschaften von Standard- und AppExpert-Zieltypen verändern; die Eigenschaften von Source Pools können hingegen nicht verändert werden.

Um die Eigenschaften eines Ziels zu verändern:

1. Klicken Sie mit der rechten Maustaste auf den Zielknoten im Projektfenster (oder markieren Sie ihn und drücken Sie [*Alt*][*F10*]) und wählen Sie dann TargetExpert aus dem lokalen Menü, um das *TargetExpert*-Dialogfenster zu öffnen.

 Hinweis Die Zieldateifelder sind Untergruppen zu den Feldern im Dialogfenster *Neues Ziel*. Eine Beschreibung dieser Felder finden Sie unter »Setzen von Zieloptionen mit dem Dialogfenster *Neues Ziel*«.

2. Aktualisieren Sie die Zieleigenschaften und wählen Sie OK zur Bestätigung.

Verschieben von Knoten in einem Projekt

Sie können Knoten auf folgende Weise verschieben:

- Ziehen Sie den Knoten mit der Maus an die gewünschte Position.
- Markieren Sie den Knoten und drücken Sie [*Alt*] und die Pfeiltasten. Der markierte Knoten kann so innerhalb der *sichtbaren* Knoten auf- und ab-

wärts bewegt werden. Unter Verwendung von [*Alt*] und der rechten und linken Pfeiltaste können Sie den Knoten innerhalb der Abhängigkeitskette vor und zurück bewegen. Haben Sie beispielsweise eine .CPP-Datei, die von einer .H-Datei abhängt (die .H-datei wird rechts versetzt unter der .CPP-Datei angezeigt), so können Sie die hierzu anzuwählende Header-Datei auf dieselbe Ebene wie die .CPP-Datei verschieben, indem Sie [*Alt*][*Links*] drücken.

Knoten kopieren

Knoten einer Projektdatei können als Duplikat oder Referenz kopiert werden. Wenn Sie einen Knoten als Duplikat kopieren, plaziert der Projekt-Manager eine identische, aber *unabhängige* Kopie des Knotens an den gewählten Ort. Die kopierten Knoten erben alle Eigenschaften des Originalknotens; Sie haben die Möglichkeit, jede einzelne dieser Eigenschaften zu verändern.

Als Referenz zu kopieren bedeutet, von einer anderen Stelle im Projekt aus Bezug auf den Knoten zu nehmen. Die Referenzkopie unterscheidet sich nicht vom Originalknoten. Verändern Sie die Struktur des Originalknotens, so wird die Referenzkopie ebenfalls verändert. Eine Referenzkopie erbt jedoch nicht die Optionen des Originals; Sie können in der Kopie ohne Beeinflussung des Originalknotens Eigenschaften verändern und Optionen neu definieren.

Ein Duplikat wird folgendermaßen erstellt:

1. Wählen Sie den/die Knoten, die sie kopieren wollen ([*Umschalt*] oder [*Strg*] und Mausklick, um mehrere Knoten anzuwählen). Hat ein Knoten Bestimmende Knoten, so werden diese automatisch kopiert und brauchen nicht extra angewählt zu werden.
2. Halten Sie die [*Strg*]-Taste gedrückt und ziehen Sie die ausgewählten Knoten an die gewünschte Position.
3. Die Duplikate erscheinen, sobald Sie die Maustaste loslassen. Nehmen Sie jetzt Änderungen am Original vor, so haben diese keine Auswirkungen auf das Duplikat.

Eine Referenz wird wie folgt angelegt:

1. Wählen Sie den Knoten, für den Sie eine Referenz wünschen, an. Auch hier sind die Bestimmenden automatisch eingeschlossen.
2. Halten Sie [*Alt*] gedrückt und ziehen Sie den angewählten Knoten zur gewünschten Position.
3. Die Kopie erscheint, sobald Sie die Maustaste loslassen. Die Referenz wird in *hellerer* Schrift angezeigt, um sie vom Duplikat zu unterscheiden und Sie daran zu erinnern, daß Änderungen am Original (wie

das Entfernen oder Hinzufügen von Bestimmenden) sich auf *alle* Referenzen auswirken.

Achtung!

Sie können als Referenz kopierte Knoten nicht hinzufügen, löschen oder verändern; um diese Art von Knoten zu verändern, müssen Sie die Hauptkopie bearbeiten. Löschen Sie ein Original, so verschwinden auch *alle* Referenzen. Sie können diesen Vorgang nicht rückgängig machen.

Umwandeln von Projektdateien in Make-Dateien

Mit der IDE können Sie Projektdateien (.IDE-Dateien) in Make-Dateien (.MAK-Dateien) konvertieren. Um ein Projekt in eine Make-Datei zu konvertieren:

1. Öffnen Sie die zu konvertierende Projektdatei.
2. Wählen Sie *Projekt/Make-Datei erzeugen*. Die IDE erzeugt eine Make-Datei mit demselben Namen wie das Projekt, aber der Erweiterung .MAK und legt sie im Editierpuffer ab. Die IDE zeigt die neue Make-Datei in einem Editierfenster an.
3. Wählen Sie *Datei/Speichern* zum Speichern der neuen Make-Datei.

Importieren von Projekten aus Borland C++ 3.x

Der Projekt-Manager kann Projekte aus früheren Borland und Turbo C++ Versionen laden. Wählen Sie *Projekt/Projekt Öffnen* und geben Sie den Namen der .PRJ-Datei ein. Um die Projektdateien aus den Versionen 3.0 und 3.1 in der Dateinamenliste anzuzeigen, ändern Sie die zu suchenden Erweiterungen im Dropdown-Fenster *Dateityp* von *.IDE nach *.PRJ.

Wenn Sie eine alte Projektdatei öffnen, konvertiert sie der Projekt-Manager in eine neue. Denken Sie daran, das neue Projekt zu speichern, wenn Sie es weiter unter dieser Version von Borland C++ verwenden wollen. Wählen Sie, um das Projekt zu speichern, *Optionen/Speichern*. Ist das Projekt markiert, klicken Sie auf OK. Das neue Projekt wird mit der Namenserweiterung .IDE gespeichert.

Anpassen des Projektfensters

Vorgabemäßig erscheinen im Projektfenster Ziel- und bestimmende Knoten. Die Anzeige von Knoten und Optionen können Sie folgendermaßen bestimmen:

1. Wählen Sie *Optionen/Umgebung*, um das Dialogfenster *Umgebungsoptionen* zu öffnen und wählen Sie *Projektansicht*. Auf der rechten Seite des Fensters erscheinen die Optionen für die Projektansicht.

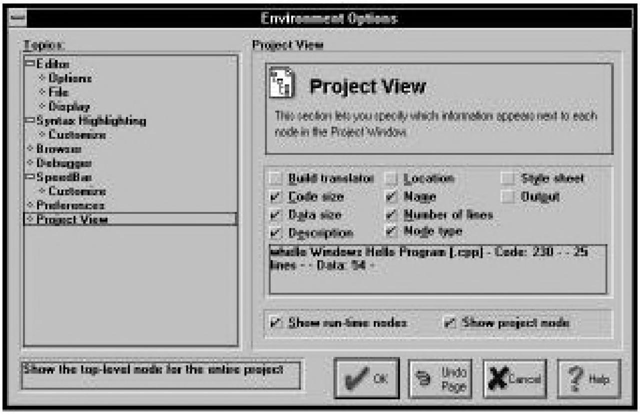


Abbildung 2.3:
Das Dialogfenster
Umgebungsoptionen

2. Setzen oder löschen Sie Markierungen an den gewünschten Optionen. Ein Beispielknoten namens WHELLO ändert sich mit den von Ihnen vorgenommenen Einstellungen. Hier können Sie sehen, in welcher Form sämtliche Knoten im Projektfenster erscheinen.

 - **Übersetzer** zeigt den auf den Knoten angewandten Übersetzer an.
 - **Code-Größe** zeigt die Gesamtgröße aller Code-Segmente in Bytes an. Diese Information erscheint erst nach Compilierung des Knotens.
 - **Datengröße** zeigt die Größe des Datensegments in Bytes an. Diese Information erscheint erst nach Compilierung des Knotens.
 - **Beschreibung** zeigt eine optionale Beschreibung des Knotens im Projektbaum an. Die Beschreibung können Sie unter Verwendung des Dialogfensters *Knotenattribute bearbeiten* im lokalen Menü des Projekt-Managers eingeben.
 - **Ort** listet den Pfad auf, in dem sich die mit dem Knoten verbundene Quelldatei befindet.
 - **Name** zeigt den Namen des Knotens an.
 - **Zeilenzahl** zeigt die Anzahl der Zeilen in der mit dem Knoten verbundenen Datei an (erscheint erst nach Compilierung des Knotens).
 - **Knotentyp** beschreibt den Typ des Knotens (z.B. .cpp oder .c).
 - **Schablone** nennt die mit dem Knoten verbundene Schablone.
 - **Ausgabe** nennt Pfad und Dateinamen, die bei der Übersetzung des Knotens erzeugt werden. Ein .CPP-Knoten beispielsweise erzeugt eine .OBJ-Datei.
 - **Laufzeitknoten anzeigen** zeigt die Knoten an, die der Projekt-Manager beim Aufbau des Projekts verwendet. Diese Option listet z.B. Startup Code und Bibliotheken auf.

- **Projektknoten anzeigen** zeigt den Projektknoten an, von dem alle Ziele abhängen.

3. Klicken Sie auf OK, um das Dialogfenster *Umgebungsoptionen* zu schließen.

4 Um Ihre Projektanpassungen zu speichern, wählen Sie *Optionen/Speichern* und dann *Projekt*. Für die verschiedenen Projekte, an denen Sie arbeiten, können Sie unterschiedliche Optionssätze speichern.

Gruppieren von Dateien mit Quelltextsammlungen

Eine *Quelltextsammlung* ist eine Zusammenfassung bestimmender Knoten, auf die durch mehrere Zielknoten verwiesen werden kann. Wenn durch einen Zielknoten auf eine Quelltextsammlung verwiesen wird, übernehmen deren Knoten die Optionen und Zielattribute des Ziels. Da Quelltextsammlungen Ihnen ermöglichen, verschiedene Ziele unter Verwendung einer Menge gemeinsamer Knoten zu erzeugen, ist die Handhabung der von den Zielen verwendeten Dateien einfach. Mit Quelltextsammlungen können Sie beispielsweise sowohl 16- als auch 32-Bit-Anwendungen unter Benutzung einer einzelnen Menge von bestimmenden Knoten erstellen. So müssen Sie sich beim Hinzufügen oder Löschen einer Datei der Quelltextsammlung nicht um die Aktualisierung sämtlicher Zielknoten kümmern; durch den Verweis auf die Quelltextsammlung werden sie automatisch aktualisiert.

Quelltextsammlungen können Sie auch verwenden, wenn Sie mehrere Header-Dateien haben, die im Verlauf des Projektaufbaus eingebunden werden sollen. Wenn Sie die Header-Dateien in eine Quelltextsammlung aufnehmen, können Sie auf sie verweisen, wann immer sie benötigt werden. Sollten Änderungen an der Gruppe der Header-Dateien notwendig werden, müssen Sie nur die ursprüngliche Quelltextsammlung aktualisieren. Fügen Sie der Quelltextsammlung eine neue Header-Datei hinzu, werden alle Referenzkopien automatisch aktualisiert.

Quelltextsammlungen sind auch nützlich, um die gleiche Schablone verschiedenen Knoten zuzuordnen. Haben Sie also drei Ziele und wollen für diese denselben Schablone verwenden, können Sie die Quelltextsammlung verweisen, die die Schablone enthält, anstatt jedem Knoten die gleiche Schablone einzeln hinzuzufügen. Soll die Schablone aktualisiert werden (z.B. wenn vom Compilieren mit zum Compilieren ohne Debug-Informationen gewechselt werden soll), können Sie alle Ziele durch Veränderung der einzelnen Schablone aktualisieren. Quelltextsammlungen können Sie auch verwenden, um selbstdefinierte Hilfsprogramme auf Projektknoten anzuwenden. Weitere Informationen finden Sie im Beispielprojekt \BC45\EXAMPLES\IDE\DELIVER.

Erstellen einer Quelltextsammlung

Wenn Sie eine Quelltextsammlung erstellen, erzeugen Sie einen Zielknoten mit einer Gruppe von Knoten darunter. Der Zielknoten der Quelltextsammlung kann jedoch nicht compiliert werden – um die Knoten in einer Quelltextsammlung zu compilieren, müssen Sie letztere an einen anderen Zielknoten kopieren. Quelltextsammlungen arbeiten am vorteilhaftesten, wenn Sie sie als Referenz kopieren.

1. Erzeugen Sie durch Anwählen von *Projekt/Neues Ziel* einen neuen Zielknoten in Ihrem Projekt.
2. Geben Sie im Feld *Zielname* den Namen der Quelltextsammlung ein.
3. Wählen Sie den Typ der Quelltextsammlung aus der Liste und drücken Sie OK, um einen Zielknoten für eine Quelltextsammlung in Ihrem Projekt zu erzeugen.
4. Wählen Sie den Knoten der Quelltextsammlung im Projektbaum und drücken Sie [*Einfg*], um das Dialogfenster *Zu Projektliste hinzufügen* zu öffnen.
5. Wählen Sie die gewünschten Quelldateien und drücken Sie OK, um diese der Quelltextsammlung hinzuzufügen.
6. Erstellen Sie eine Referenzkopie der Quelltextsammlung, indem Sie die [*Alt*]-Taste gedrückt halten und die Quelltextsammlung auf die gewünschten Zielknoten ziehen.

Hinweis

Ein einsatzfähiges Beispielprojekt für eine Quelltextsammlung ist die Datei SRCPOOL.EXE im Verzeichnis \BC45\EXAMPLES\IDE\SRCPOOL. Die Projektdatei enthält eine Textdatei, die die Verwendung der Quelltextsammlung im Beispiel beschreibt.

Übersetzer, Viewer und Hilfsprogramme

Übersetzer, Viewer und Hilfsprogramme sind interne und externe Programme, die Ihnen über die IDE zur Verfügung stehen.

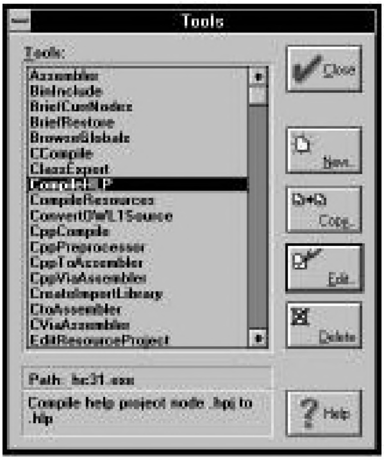
- *Übersetzer* sind Programme, die einen Dateityp in einen anderen überführen. Der C++ Compiler ist beispielsweise ein Übersetzer, der .OBJ- aus .CPP-Dateien erzeugt. Der Linker wiederum übersetzt .OBJ-, .LIB-, .DEF- und .RES-Dateien in .EXE-Dateien.
- *Viewer* sind Programme, mit denen Sie den Inhalt eines ausgewählten Knotens ansehen können. Ein Editor beispielsweise ist ein Viewer, über den Sie den Quell-Code einer .CPP-Datei untersuchen können. Resource Workshop ist ein Viewer für Windows-Ressourcendateien.

- *Hilfsprogramme* helfen Ihnen beim Erzeugen und Testen Ihrer Anwendungen. Beispiele hierfür sind der Turbo Debugger und GREP.

Abhängig von der Namenserweiterung des Knotens verbindet die IDE jeden Knoten eines Projekts mit verschiedenen Übersetzern oder Viewern. Jeder Knoten ist, obwohl er mehreren unterschiedlichen Übersetzern oder Viewern zugeordnet werden kann, standardmäßig einem *einzelnen* Übersetzer oder Viewer zugeordnet. So erkennt die IDE, daß das Editierfenster geöffnet werden soll, wenn Sie einen .CPP-Knoten anklicken (Doppelklicken auf einen Knoten ruft dessen Vorgabe-Viewer auf). Um zu sehen, welcher Knotentyp (abhängig von der Dateinamenerweiterung) für einen bestimmten Übersetzer oder Viewer voreingestellt ist:

1. Wählen Sie *Optionen/Hilfsprogramme*, um das Dialogfenster *Hilfsprogramme* zu öffnen.

Abbildung 2.4: Das Dialogfenster *Hilfsprogramme*



2. Wählen Sie das zu untersuchende Element aus der Liste.
3. Mit *Edit* gelangen Sie in das Dialogfenster *Hilfsprogramm-Optionen*.
4. Wählen Sie *Erweitert*, um das Fenster *Erweiterte Hilfsprogramm-Optionen* zu öffnen und schauen Sie sich das Textfeld *Vorgabe für* an.

Manche Knoten haben einen *Extras*-Befehl für das lokale Menü, das Sie durch Klicken mit der rechten Maustaste auf den Knoten aktivieren können. Dieser Befehl listet die für den gewählten Knotentyp alternativ zur Verfügung stehenden Übersetzer auf. Im Menü *Extras* für einen .CPP-Knoten beispielsweise erscheinen die Befehle *C an Assembler*, *C++ an Assembler* und *Präprozessor*. Haben Sie einen .DLL-Knoten angewählt, so erscheint der Befehl *Implib*. Mit dem Befehl *Extras* können Sie jeden für einen gewählten Knotentyp verfügbaren Übersetzer aufrufen. Durch Anwählen eines bestimmenden Knotens im Projektbaum und Aufruf von

Knotenattribute bearbeiten im lokalen Menü können Sie dem Knoten wieder seinen standardmäßigen Übersetzer zuweisen.

Hinzufügen von Übersetzern, Viewern und Hilfsprogrammen zur IDE

Das Dialogfenster *Hilfsprogramme* zeigt die voreingestellten Übersetzer, Hilfsprogramme und Viewer. Die folgenden Schritte beschreiben, wie man dieser Programmliste ein Element hinzufügt.

1. Mit *Optionen/Hilfsprogramme* gelangen Sie in das Dialogfenster *Hilfsprogramme*. Hier finden Sie eine Liste der Standardübersetzer, -Viewer und -Hilfsprogramme.
2. Wählen Sie *Neu*, um der Liste ein neues Programm hinzuzufügen (um ein bereits aufgeführtes Programm zu verändern, klicken Sie es an und wählen Sie *Bearbeiten*).
3. Stellen Sie die folgenden Optionen in dem Dialogfenster *Optionen/Hilfsprogramme*:
 - **Name** ist eine Beschreibung des Elements, das Sie hinzufügen. Er erscheint in der Liste der Hilfsprogramme.
 - **Pfad** nennt Pfad und Namen des ausführbaren Programms. Diese Auswahl können Sie unter Verwendung des Symbols zum Durchsuchen vervollständigen.
 - **Kommandozeile** enthält alle beliebigen Kommandozeilenoptionen, Transfermakros und IDE-Filter, die Sie an das Programm übergeben möchten. Informationen zu Transfermakros finden Sie in der Online-Hilfe (wenn Sie mit Transfermakros experimentieren wollen, versuchen Sie es mit **$prompt**). IDE-Filter sind .DLL-Dateien, die Schnittstellen zwischen Hilfsprogrammen und der IDE erzeugen (das Hilfsprogramm *GrepFile* beispielsweise verwendet einen Filter, um einen Text im Meldungsfenster auszugeben). Weitere Informationen zu Filtern finden Sie im Projekt FILTER.IDE im Verzeichnis \BC45\EXAMPLES\IDE\FILTER. Um die verwendeten Transfermakros und Filter zu sehen, wählen Sie zunächst *Optionen/Hilfsprogramme*, dann *GrepFiles* und *Bearbeiten*.
 - **Menütext** erscheint in lokalen Menüs und im Menü *Hilfsprogramme*. Wollen Sie Ihrem Menütext ein Tastenkürzel zuweisen, setzen Sie dem Abkürzungsbuchstaben ein kaufmännisches Und (&) voran; dann erscheint dieser Buchstabe unterstrichen im Menü. Mit `&File` z.B. wird F zum Tastenkürzel für File erklärt. Soll im Menütext ein kaufmännisches Und erscheinen, so verwenden Sie dieses Symbol zweimal (`&&Up&date` erscheint im Menü als `&Update`)

Hinweis

Soll das Programmelement im lokalen oder im *Hilfsprogramm*-Menü erscheinen, *müssen* Sie hier Text eingeben.

- **Hinweis** ist erläuternder Text, der in der Statuszeile des Dialogfensters *Hilfsprogramme* erscheint, wenn Sie das Programmelement auswählen.

4. Öffnen Sie das Dialogfeld *Erweiterte Optionen* (wählen Sie *Erweitert*), um Optionen für Ihr neues Programm zu setzen. Abhängig vom Typ des gewählten Hilfsprogramms (Einfacher Transfer, Übersetzer oder Viewer) stehen unterschiedliche Felder zur Verfügung. Wenn Sie einen Übersetzer wählen, verfügt das Programm über Make- und Build-Prozesse.

 - **Zum Menü *Hilfsprogramme*** fügt das Element dem Menü *Hilfsprogramme* hinzu.
 - **Zum lokalen Menü** fügt den vebundenen lokalen Menüs einen Viewer oder Übersetzer hinzu.
 - **Zielübersetzer** steht für Übersetzer und Viewer zur Verfügung. Bei Übersetzern bestimmt dieses Feld, ob das Programm ein endgültiges Ziel (wie z.B. eine .EXE-Datei) oder eine Zwischendatei (etwa eine .OBJ- oder .I-Datei) erstellt. Wenn Sie dieses Feld markieren, erzeugt der Übersetzer eine endgültige Zieldatei, die in dem im Textfeld *Endgültig* bestimmten Verzeichnis abgespeichert wird (wählen Sie hierzu *Projekt/Optionen/Verzeichnisse*). Wenn Sie *Zielübersetzer* nicht markieren, wird die übersetzte Datei in dem im Textfeld *Zwischendatei* angegebenen Verzeichnis abgelegt.

 Bei Viewern legt *Zielübersetzer* fest, daß der Viewer nur auf bereits übersetzte Knoten (wie .OBJ- oder .EXE-Dateien) angewendet wird; ein Knoten muß also übersetzt sein, bevor Sie ihn ansehen können.
 - **Übersetzen von** bestimmt die Knotentypen (abhängig von der Namenserweiterung), die ein Übersetzer übersetzen kann. Möchten Sie mehrere Knotentypen festlegen, so geben Sie die Erweiterungen durch Semikola getrennt ein.

 Wenn Sie in diesem Feld eine Dateinamenerweiterung eingeben, fügt der Projekt-Manager den Übersetzer dem Menü *Extras* den Projektknoten hinzu, die diese Erweiterung haben. Es ist allerdings wichtig, daß jeder Knotentyp nur einen einzigen Standardübersetzer haben kann (s.u., Option **Standard**).

 Wie dies funktioniert, können Sie im Hilfsprogramm CppCompile nachsehen (wählen Sie *Optionen/Hilfsprogramme*, doppelklicken Sie auf *CppCompile* und klicken Sie dann auf *Erweitert*). Das Dialogfeld *Erweiterte Hilfsprogramm-Optionen* zeigt, daß der C++ Compiler ein Übersetzer für .CPP-, .C-, .CAS- und .H-Dateien ist. Trägt einer Ihrer bestimmenden Knoten die Erweiterung .C, so erscheint CppCompile im Menü *Extras*, wenn Sie den Knoten mit der rechten Maustaste anklicken und *Extras* wählen.

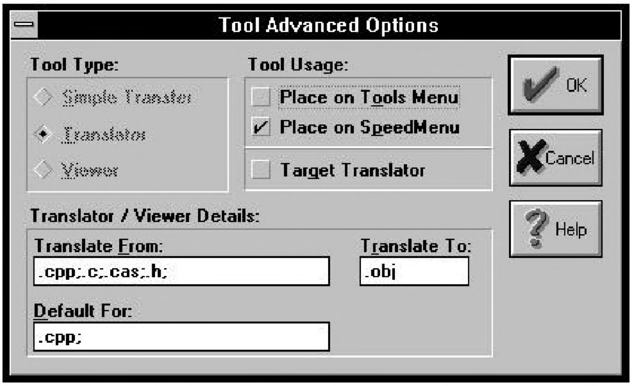


Abbildung 2.5: Das Dialogfeld *Erweiterte Hilfsprogramm-Optionen*

- **Übersetzen nach** bestimmt die Namenserweiterung der vom Übersetzer erzeugten Datei. HC31.EXE beispielsweise konvertiert .HPJ- zu .HLP-Dateien, daher befindet sich im Feld die Erweiterung `.hlp`.
- **Anwenden auf** ähnelt dem Feld **Übersetzen von**, nur wird es für Viewer anstatt für Übersetzer verwendet.
- **Standard** ändert den Standardübersetzer oder -viewer der IDE in den von Ihnen bestimmten Dateityp. Geben Sie die Namenserweiterungen (getrennt durch Semikola) für diejenigen Dateitypen an, deren Voreinstellung Sie außer Kraft setzen wollen.

5. Wählen Sie zweimal OK, um Ihre Einstellungen zu bestätigen und schließen Sie das Dialogfeld.

Ihr neues Hilfsprogramm wurde nun der Hilfsprogrammliste des verbundenen Projekts hinzugefügt, ferner dem Menü *Hilfsprogramme* oder dem lokalen Menü, abhängig von Ihrer Auswahl. Wenn Sie das Element dem Menü *Hilfsprogramm* hinzugefügt haben, können Sie dies überprüfen, indem Sie im Hauptmenü *Hilfsprogramm* wählen; der neue Programmname erscheint in der Liste der Hilfsprogramme.

Hinweis

Obwohl Sie über den Projekt-Manager Ihre eigenen Hilfsprogramm-Einheiten bestimmen können, finden diese nur bei dem Projekt Anwendung, dem Sie diese hinzugefügt haben und werden nicht als ständige Bestandteile der IDE behandelt. Übersetzer, Viewer und Hilfsprogramme können jedoch an neue und bestehende Projekte *übergeben* werden. Weitere Informationen zur gemeinsamen Nutzung externer Programme durch mehrere Projekte finden Sie unter »Mehrfachbenutzung von Schablonen«.

K A P I T E L 3

Bestimmen von Projektoptionen und Compilierung

Nachdem Sie eine Projektdatei erzeugt und den Programm-Code für die bestimmenden Knoten in Ihr Projekt eingegeben haben, müssen Sie Optionen für die verschiedenen Projektknoten setzen, bevor Sie das Projekt compilieren können. Dieses Kapitel beschreibt, wie Optionen in einem Projekt festgelegt werden, wie Sie diese ansehen können, wie ein Projekt compiliert wird und wie das Fenster *Meldung* zum Anzeigen und Beheben von Compilierzeit-Fehlern verwendet wird. Zusätzlich finden Sie eine vollständige Übersicht der Compiler- und Linker-Optionen, die von der IDE aus eingestellt werden können.

Einstellen der Projektoptionen

Projektoptionen teilen der IDE mit, wie die Knoten Ihres Projekts zu compilieren und zu linken sind, um die benötigten Ziele zu bilden. Solche Optionen legen beispielsweise fest, ob Sie eine Anwendung mit oder ohne Debug-Informationen erzeugen wollen, wo nach Quell-Code gesucht werden soll, welche Arten von Compiler-Optimierungen Sie benutzen möchten usw.

Mit dem Projekt-Manager können Sie Projektoptionen auf zwei Arten festlegen:

- Sie können den Projektknoten Schablonen zuordnen.
- Sie können die Einstellungen in einer Schablone unter Verwendung lokaler Optionen überschreiben.

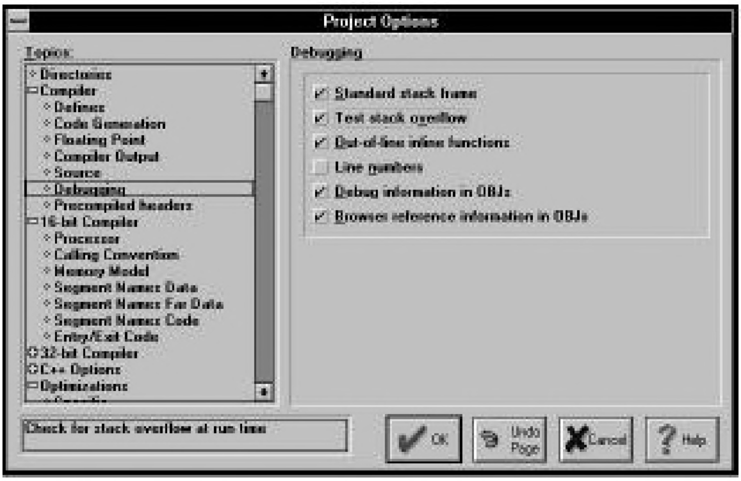
Schablonen gruppieren eine Menge von Optionseinstellungen zu einer Einheit. Wenn Sie eine Schablone erzeugt haben, können Sie sie einem Knoten, einer Gruppe von Knoten oder einem vollständigen Projekt zuordnen. *Überschreibende lokale Optionen* sind Einstellungen, die Vorrang vor den Schabloneneinstellungen auf Knotenebene haben.

Benutzen von Schablonen

Oftmals erfordern unterschiedliche Projektknoten unterschiedliche Optionseinstellungen. So möchten Sie vielleicht mit einem Optionssatz .C-Dateien und mit einem anderen .CPP-Dateien compilieren, oder Sie möchten ein Ziel mit, das andere ohne Debug-Informationen compilieren. Indem Sie verschiedenen Knoten Ihres Projekts verschiedene Schablonen zuordnen, können Sie den Aufbau der verschiedenen Knoten leicht kontrollieren. Zusätzlich erleichtern Schablonen die Ansicht und Pflege der Projektoptions-Einstellungen.

Um die Optionen zu sehen, die in eine Schablone aufgenommen werden können, öffnen Sie über *Optionen/Projekt* das Dialogfenster *Projektoptionen*. Dieses Fenster enthält auf der linken Seite eine hierarchische Themenliste mit einer Auflistung der zu jedem Thema verfügbaren Optionen auf der rechten Seite. Zum Ein- und Ausblenden der Themenliste klicken Sie auf die Symbole + und - auf der linken Seite der Themenliste.

Abbildung 3.1: Das Dialogfenster *Projektoptionen*



Ein Beispiel zur Verwendung von Schablonen ist die Datei STYLESHT.IDE im Borland C++ Verzeichnis EXAMPLES\IDE\STYLESHT. Für jede seiner beiden Versionen von WHELLO verwendet diese Datei eine andere Schablone. Das Projekt enthält ferner eine die Verwendung von Schablonen erläuternde Textdatei.

Vordefinierte Schablonen

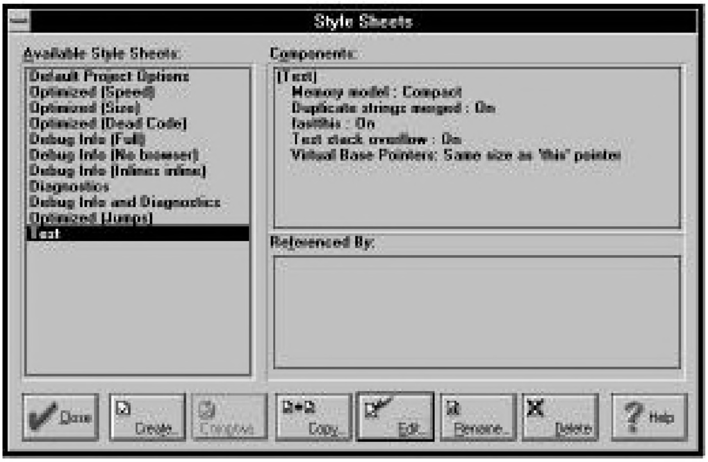
Der Projekt-Manager enthält mehrere vordefinierte Schablonen, die Sie jedem Knoten Ihres Projekts zuordnen oder aber entsprechend den Projekterfordernissen anpassen können, wie unter »Verwalten von Schablonen« beschrieben.

Wählen Sie *Optionen/Schablonen*, um die vordefinierten Schablonen zu sehen. Es öffnet sich das Dialogfenster *Schablonen* (Abbildung 3.2) mit einer Liste der vordefinierten Schablonen auf der linken und einer Beschreibung der ausgewählten Schablone auf der rechten Seite.

Standard-Projektoptionen

Beim ersten Erstellen eines Projekts vererbt dieses die Schablone namens *Standard-Projektoptionen* (weitere Informationen zu diesen Voreinstellungen finden Sie unter »Standard-Optionen«). Wenn Sie alle Bestandteile Ihres Projekts mit denselben Optionen erzeugen können, haben Sie die Möglichkeit, diese Schablone über das Dialogfenster *Projektoptionen* zu bearbeiten. Wenn jedoch verschiedene Knoten Ihres Projekts unterschiedliche Optionseinstellungen erfordern, sollten Sie die Standard-Optionen überschreiben, indem Sie den Knoten Ihres Projekts verschiedene Schablonen zuordnen.

Abbildung 3.2: Dialogfenster *Schablonen*



Achtung!

Seien Sie vorsichtig beim Verwenden des Befehls *Optionen/Projekt*. Wenn Ihr Projekt mehr als einen Zielknoten enthält, bewirken vorgenommene Veränderungen *stets* auch eine Änderung der Standard-Optionen des Projekts (ganz gleich, welchen Knoten Sie beim Wählen des Befehls markiert haben). Daher erben *alle* Ziele des Projekts die Veränderungen, die Sie vornehmen, wenn Sie den Befehl *Projekt/Optionen* wählen. Wenn Sie Projektoptionen verändern, ohne daß ein Projekt geladen ist, bewirken diese Änderungen außerdem eine Aktualisierung der Schablone *Standard-Projektop-*

tionen; alle später erzeugten Projekte erben dann diese neuen Voreinstellungen. Müssen Sie zu den werksmäßigen Voreinstellungen zurückkehren, löschen Sie die Datei BCWDEF.BCW (im Verzeichnis \BC45\BIN) und öffnen und schließen Sie dann die IDE, um eine neue Datei zu erstellen.

Schablonen verwalten

Über die Schaltfelder am unteren Rand des Dialogfelds *Schablonen* können Sie benutzerdefinierte Schablonen erzeugen, zusammenstellen, kopieren, bearbeiten, umbenennen und löschen.

- Mit **Erzeugen** entwerfen Sie eine neue Schablone für das aktuelle Projekt. Um eine Schablone zu erzeugen:

1. Wählen Sie die Schaltfläche *Erzeugen* und geben Sie einen Namen für die neue Schablone in das Dialogfeld *Schablone erzeugen* ein. Wählen Sie OK, um sie der Liste *Verfügbare Schablonen* hinzuzufügen.
2. Wählen Sie die neue Schablone aus dieser Liste und dann die Schaltfläche *Zusammenstellen*, *Kopieren* oder *Bearbeiten*, um eine benutzerdefinierte Schablone zu erzeugen.

- Mit **Zusammenstellen** erzeugen Sie eine Schablone, die eine Kombination von Optionen einer oder mehrerer Schablonen enthält. Um eine Schablone zusammenzustellen,

1. Erzeugen Sie eine neue Schablone mit dem Schalter *Erzeugen*.
2. Wählen Sie die neue Schablone aus der Liste *Verfügbare Schablonen* und klicken Sie auf *Zusammenstellen*.
3. Markieren Sie die in Ihre neue Schablone einzubindende Schablone in der Liste und verschieben Sie sie von dort in die Liste *Zusammengestellte Schablonen* (durch Doppelklicken auf die Schablone oder Anklicken des Schalters –>; entfernen Sie eine markierte Schablone aus der Liste *Zusammengestellte Schablonen* durch Klicken auf <–).
4. Fahren Sie so mit den Änderungen fort und wählen Sie OK, wenn Sie fertig sind.

Hinweis

Sie können zwar die Optionseinstellungen einer zusammengestellten Schablone nicht bearbeiten, jedoch die der darin enthaltenen Schablonen. Dies wirkt sich dann auch auf die zusammengestellte Schablone aus.

- Mit **Kopieren** erzeugen Sie eine neue Schablone aus einer vorhandenen. Wählen Sie diesen Menübefehl, werden Sie nach dem Namen der neuen Schablone gefragt. Geben Sie ihn ein und wählen Sie OK, und Sie erhalten eine genaue Kopie der ausgewählten Schablone. Dies ist eine schnelle Methode zur Erstellung einer Schablone, die einer anderen stark ähnelt. Sie brauchen nur noch die gewünschten Optionen zu verändern.

- Mit **Bearbeiten** verändern Sie die Optionseinstellungen jeder bereits existierenden Schablone einschließlich aller vordefinierten.
- Mit **Umbenennen** benennen Sie eine ausgewählte Schablone um.
- Mit **Löschen** entfernen Sie eine Schablone (dieser Vorgang kann nicht rückgängig gemacht werden).

Verbinden einer Schablone mit einem Knoten

Um eine Schablone mit einem Knoten zu verbinden:

1. Klicken Sie mit der rechten Maustaste auf den Knoten im Projektbaum (oder markieren Sie ihn und drücken Sie [*Alt*][*F10*]).
2. Wählen Sie *Knotenattribute bearbeiten* aus dem lokalen Menü, um das Dialogfenster *Knotenattribute* zu öffnen.
3. Wählen Sie eine Schablone aus der Auswahlliste und klicken Sie auf OK.

Wenn Sie eine Schablone mit einem Knoten verbinden, erben alle seine untergeordneten Knoten die Eigenschaften der gewählten Schablone. Um die Eigenschaften eines untergeordneten Knotens zu verändern, verbinden Sie ihn mit einer anderen Schablone oder verwenden Sie überschreibende lokale Optionen.

Hinweis

Obwohl Sie einem Projektknoten nur jeweils eine Schablone zuordnen können, kann eine Schablone aus mehreren verschiedenen Schablonen bestehen.

Mehrfache Benutzung von Schablonen

Wenn Sie eine benutzterdefinierte Schablone erstellen, bleibt diese mit dem Projekt verbunden, für das sie erstellt wurde und wird nicht der Liste der vordefinierten Schablonen hinzugefügt. Möchten Sie jedoch, daß ein neues Projekt von Ihnen definierte Schablonen oder Hilfsprogramme verwendet, können Sie einem Projekt die Einstellungen eines anderen vererben.

Ehe dies geschehen kann, müssen Sie die Datei BCW.INI in Ihrem Windows-Verzeichnis modifizieren. Enthält die Datei keine Einstellung für die Vererbung, müssen Sie die Einstellungen wie folgt vornehmen:

```
[Project]
;Um Standard-Projekteinstellungen an neue Projekte zu vererben (Vorgabe):
inherit=0

;Um die Einstellungen des aktuellen Projekts an neue Projekte zu vererben:
inherit=1

;Um werksmäßige Voreinstellungen an neue Projekte zu vererben:
inherit=2
```

So übergeben Sie Schablonen oder benutzerdefinierte Hilfsprogramme von einem Projekt an ein neues:

1. Modifizieren Sie BCW.INI, indem Sie inherit=1 setzen.
2. Öffnen Sie das die Schablone oder die Hilfsprogramme enthaltende Projekt, das Sie mitbenutzen wollen.
3. Wählen Sie *Projekt/Neues Projekt*.

Wenn das neue Projekt erzeugt wird, erbt es die Schablonen und benutzerdefinierten Hilfsprogramme des Projekts, das bei der Auswahl von *Projekt/Neues Projekt* geöffnet war.

Dateien der Projektbeschreibungssprache

Sie können Dateien der Projektbeschreibungssprache (.PDL) verwenden, um Schablonen und Hilfsprogramme zwischen den Projekten zu teilen. Wenn Sie ein Projekt speichern, können Sie die IDE anweisen, eine .PDL-Datei mit dem gleichen Namen wie den der Projektdatei zu erzeugen. Ebenso können Sie die IDE beim Öffnen eines Projekts anweisen, dessen .PDL-Datei zu lesen. Eine .PDL-Datei enthält Informationen über die in einem Projekt verwendeten Schablonen und Hilfsprogramme. Sie können sie bearbeiten und ihr Informationen zu den Schablonen und Hilfsprogrammen Ihrer Wahl hinzufügen. Gleichwohl sollten Sie beim Bearbeiten einer .PDL-Datei Vorsicht walten lassen, damit sie nicht – was schnell geschehen kann – unleserlich für den Projekt-Manager wird.

Wenn Sie beabsichtigen, .PDL-Dateien für die mehrfache Benutzung von Schablonen und Hilfsprogrammen zu verwenden, müssen Sie zunächst sicherstellen, daß die IDE diese Dateien erzeugen und lesen kann. Öffnen Sie hierzu die Datei BCW.INI (im Windows-Stammverzeichnis) und fügen Sie folgende Einstellungen zum Bereich [Project] der .INI-Datei hinzu:

```
[Project]
saveastext=1
readastext=1
```

Die Einstellung **saveastext** weist die IDE an, beim Speichern eines Projekts stets eine .PDL-Datei zu speichern. Die Einstellung **readastext** weist die IDE an, eine .IDE-Datei zu aktualisieren, wenn die mit ihr verbundene .PDL-Datei neuer ist.

Um Schablonen und benutzerdefinierte Hilfsprogramme durch mehrere Projekte zu nutzen, sind folgende Schritte erforderlich:

1. Ändern Sie die Datei BCW.INI wie bereits beschrieben.
2. Öffnen Sie das Projekt, zu dem Sie Schablonen oder Hilfsprogramme übertragen wollen und schließen Sie es dann (wählen Sie *Projekt/Projekt schließen*). Hierdurch wird eine .PDL-Datei für das Projekt erzeugt.
3. Öffnen Sie das Projekt, das die Schablonen und Dienstprogramme enthält, die gemeinsam genutzt werden sollen, und schließen Sie es wieder, um die aktuellen Änderungen in der .PDL-Datei zu speichern.

4. Verwenden Sie einen Texteditor, um die .PDL-Datei zu öffnen, die die gemeinsam zu nutzenden Schablonen und Dienstprogramme enthält (die Namen der .PDL- und der .IDE-Datei sind gleich).
5. Suchen Sie den Namen der Schablone. Haben Sie z.B. eine Schablone namens MEINSCHB erstellt, finden Sie in der .PDL-Datei einen Abschnitt, der folgendermaßen beginnt: { StyleSheet = "MEINSCHB".
6. Kopieren Sie den gesamten Text von der geöffneten bis zur geschlossenen Klammer. Wenn nötig, können Sie mehr als eine Schablone kopieren.

Hinweis

Um ein benutzerdefiniertes Hilfsprogramm gemeinsam zu nutzen, kopieren Sie den Abschnitt mit der Bezeichnung Subsystem=<tool>.

7. Öffnen Sie die .PDL-Datei, welche die Schablone erhalten soll.
8. Suchen Sie den Abschnitt, der Schablonen betrifft, und fügen Sie den kopierten Text am Ende der vorhandenen Schablonenliste ein.
9. Speichern Sie die .PDL-Datei, der die kopierte Schablone hinzugefügt wurde.
10. Öffnen Sie das Projekt, das die kopierte Schablone erhalten hat, um die Schablonen und Hilfsprogramme des Projekts über die .PDL-Datei zu aktualisieren.

Nach dem Kopieren von Schablonen ist es sinnvoll, die saveastext- und readastext-Flags in der Datei BCW.INI auf 0 zurückzusetzen. Die IDE wird somit angewiesen, nicht mehr die .PDL-Dateien zu speichern oder zu aktualisieren.

Setzen von überschreibenden lokalen Optionen

Optionseinstellungen können unter Verwendung *überschreibender lokaler Optionen* (local overrides) auf Knotenebene überschrieben werden. Überschreibende lokale Optionen sind hilfreich, wenn die Optionseinstellungen eines Knotens sich von denen seiner zugehörigen Schablone in ein oder zwei Punkten unterscheiden sollen.

Hinweis

Obwohl überschreibende lokale Optionen das Setzen von Optionen für einzelne Knoten erleichtern, haben sie den Nachteil, daß sie schwer aufzufinden sind. Während das Dialogfenster *Optionshierarchie* (erläutert unter »Ansehen von Optionen eines Projekts«) die Einstellungen zu Schablonen und überschreibenden lokalen Optionen für einen ausgewählten Knoten anzeigt, müssen Sie jeden Knoten einzeln untersuchen, um festzustellen, welche der Einstellungen überschrieben wurden. Es empfiehlt sich daher, für Knoten, die unterschiedliche Optionseinstellungen erfordern, jeweils eigene Schablonen zu verwenden und überschreibende lokale Optionen nur in besonderen Fällen zu benutzen.

Eine Optionseinstellung wird folgendermaßen überschrieben:

1. Markieren Sie den Knoten, dessen Einstellungen überschrieben werden sollen.
2. Klicken Sie den Knoten mit der rechten Maustaste an (oder drücken Sie [*Alt*][*F10*]) und wählen Sie *Lokale Optionen bearbeiten* aus dem lokalen Menü. Das Dialogfenster *Optionen* (es ähnelt dem Fenster *Projektoptionen*) erscheint und zeigt die Einstellungen für den Knoten an.
3. Wählen Sie die zu überschreibende Option. Wann immer Sie Einstellungen für eine Schablone verändern, markiert die IDE automatisch das Feld *Lokale Option überschreiben*.
4. Wählen Sie zum Bestätigen der neuen Einstellungen OK.

Achtung

Das Markierungsfeld *Lokale Option überschreiben* ist nur aktiviert, wenn eine Option zu einem Thema ausgewählt ist, ansonsten ist dieses Feld grau dargestellt. Wenn Sie eine Option markieren (durch Drücken von [*Tab*] oder indem Sie die Option anklicken und den Mauszeiger wegziehen), zeigt das Markierungsfeld *Lokale Option überschreiben* den Status der *markierten* Option. Daher müssen Sie jede Option zu einem Thema einzeln markieren, um zu sehen, welche Option lokal überschrieben wurde. Wenn Sie eine Option *auswählen* (durch Anklicken oder Markieren und Drücken der Taste [*Eingabe*]), ändern Sie deren Einstellung, wodurch das Feld *Lokale Option überschreiben* stets markiert wird.

So können Sie eine Änderung rückgängig machen:

1. Klicken Sie mit der rechten Maustaste auf den Knoten, dessen Einstellung Sie verändern wollen, und wählen Sie *Lokale Optionen bearbeiten* aus dem lokalen Menü.
2. Markieren Sie im Dialogfeld *Optionen* das Thema, das die überschriebene Einstellung enthält.

 Wenn Sie eine Themenseite markieren, die eine lokal überschriebene Option enthält, aktiviert der Projekt-Manager den Schalter *Seite rückgängig*.
3. Markieren Sie die Option (durch Drücken von [*Tab*] oder indem Sie die Option anklicken und den Mauszeiger wegziehen), dessen Überschreibung Sie rückgängig machen wollen, und das Feld *Lokale Option überschreiben* wird markiert.
4. Klicken Sie das genannte Markierungsfeld an, um die Überschreibung rückgängig zu machen; die Option wechselt zurück zur Standard-Einstellung ihrer Schablone. Um die Einstellungen des gesamten Themas auf die in der zugehörigen Schablone enthaltenen zurückzusetzen, wählen Sie den Schalter *Seite rückgängig*.
5. Bestätigen Sie die Änderungen mit OK.

Ansehen der Optionen eines Projekts

Da Sie jedem Knoten eine eigene Schablone und zusätzlich lokale Optionen zuordnen können, sollten Sie die Möglichkeit haben, einen schnellen Überblick über die Optionseinstellungen für jeden Knoten zu gewinnen.

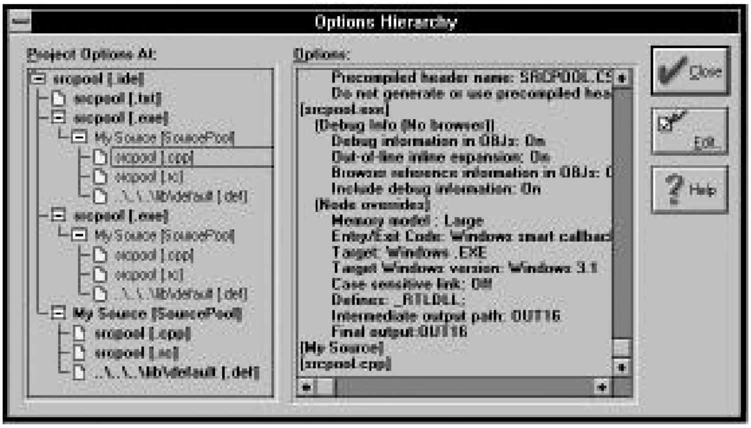
Die Optionen für die Knoten in Ihrem Projekt können Sie folgendermaßen sehen:

1. Klicken Sie mit der rechten Maustaste auf jeden beliebigen Knoten im Projektfenster und wählen Sie *Optionshierarchie anzeigen* oder klicken Sie den Schalter in der Symbolleiste an.

 Das Dialogfenster *Optionshierarchie* zeigt links eine Liste aller Knoten und die von diesen verwendeten Optionen auf der rechten Seite an. Das Ein- und Ausblenden von Knoten funktioniert wie im Projektfenster, nur daß Autodependency-Knoten nicht aufgeführt werden.

 Eine von Doppelsternen (**) umrahmte Option in der Liste zeigt an, daß diese durch einen abhängigen, weiter unten in der Optionsliste befindlichen Knoten überschrieben wurde (entweder mit einer Schablone oder mit lokalen Optionen). Die Sterne werden nur erscheinen, wenn der Knoten mit der überschriebenen Option markiert ist.

Abbildung 3.3: Das Dialogfenster *Optionshierarchie*



2. Wenn Sie einen Knoten in der Liste *Optionen zu* markieren, erscheinen dessen Einstellungen rechts in der Liste *Optionen*.

 Die Liste *Optionen* zeigt die Bestandteile des Projekts in eckigen Klammern an. Am Kopf der Liste sehen Sie den Namen des Projekts, gefolgt von den Standard-Projektoptionen. Darunter befindet sich der Name des mit dem markierten Knoten verbundenen Ziels. Hat der Knoten eine zugehörige Schablone, wird diese daneben (ebenfalls geklammert) zusammen mit ihren Einstellungen angezeigt. Sollten Sie

irgendwelche Einstellungen überschrieben haben, werden diese in der Liste unter [Überschriebene Knoten] angezeigt. Die Liste *Optionen* führt die Einstellungen für alle Vorfahren des markierten Knotens auf.

3. Wollen Sie eine Option verändern, so doppelklicken Sie auf diese oder wählen diese an und klicken auf *Bearbeiten*. Auf diese Weise bearbeitete Optionen werden zu überschreibenden lokalen Optionen.
4. Ist die Durchsicht der Knotenoptionen abgeschlossen, klicken Sie OK.

Projekte compilieren

Es gibt zwei Möglichkeiten, Projekte zu compilieren: Neu compilieren und Aktualisieren. *Neu compilieren* compiliert und linkt sämtliche Knoten eines Projekts unabhängig von der Erstellungszeit der einzelnen Dateien. *Aktualisieren* vergleicht die Zeitmarke des Zielknotens mit den Zeitmarken aller zum Aufbau verwendeten Dateien und compiliert und linkt nur die Knoten, die zur Aktualisierung des Ziels notwendig sind.

Um ein Projekt zu compilieren, öffnen Sie dieses mit *Projekt/Öffnen* und wählen Sie entweder *Compilieren*, *Aktualisieren* oder *Neu compilieren* aus dem Projektmenü (in der Symbolleiste befinden sich auch drei ähnlich aussehende, diesen Befehlen entsprechende Schalter).

- **Compilieren** ([*Alt*][*F9*]) compiliert den Code im aktiven Editorfenster. Wenn ein Projektfenster aktiv ist, werden alle ausgewählten Knoten des Projekts übersetzt. Untergeordnete Knoten werden nicht übersetzt, wenn sie nicht auch ausgewählt sind.
- **Projekt aktualisieren** ([*F9*]) übersetzt alle Knoten in einem Projekt, die seit dem letzten Compilieren verändert wurden. Wenn kein Projekt offen ist, wird die Datei im Puffer des aktiven Editorfensters compiliert.

 Wenn Sie *Projekt aktualisieren* wählen, sucht der Projekt-Manager den Projektbaum von oben nach unten ab, bis er auf einen Knoten ohne Abhängige stößt. Er vergleicht dann dessen Datum und Zeit mit Datum und Zeit des übergeordneten Knotens. Der Projekt-Manager übersetzt den untergeordneten Knoten nur, wenn er aktueller als der übergeordnete ist. Der Projekt-Manager liest nun den Projektbaum von unten nach oben, überprüft Datum und Zeit des nächsten Knotens und fährt so fort, bis alle Knoten, die seit dem letzten Compilieren verändert wurden, übersetzt sind.
- **Projekt neu compilieren** übersetzt alle Knoten in einem Projekt, auch wenn sie seit dem letzten Compilieren nicht verändert wurden, beginnend mit dem Projektknoten und alle darunter folgenden Ziele abarbeitend. Wählen Sie *Abbrechen*, um den Vorgang anzuhalten.

Wenn Sie *Projekt neu compilieren* wählen, beginnt der Projekt-Manager mit dem ersten Zielknoten und arbeitet den Projektbaum von oben nach unten ab bis zu dem Punkt, an dem er auf einen Knoten ohne Abhängigkeiten trifft. Diesen Knoten (und alle anderen auf derselben Ebene) compiliert der Projekt-Manager als ersten und bewegt sich dann wieder aufwärts im Projektbaum, wobei er alle zum Erzeugen des Ziels notwendigen Knoten compiliert und linkt. Dieser Vorgang wird in umgekehrter Richtung wiederholt, bis alle Ziele aktualisiert sind.

Haben Sie beispielsweise ein Projekt mit einem .EXE-Ziel, das von zwei einzelnen .OBJ-Dateien abhängig ist, erzeugt der Projekt-Manager die erste .OBJ-Datei, indem er all deren Abhängigkeiten compiliert; dann erzeugt er die nächste .OBJ-Datei. Wurden die Abhängigkeiten eines Zielknotens einmal erzeugt, kann der Projekt-Manager den Zielknoten compilieren oder linken. So wird er in diesem Falle die beiden .OBJ-Dateien (und alle Laufzeitknoten) linken, um die endgültige .EXE-Datei zu erzeugen.

Compilieren von Teilen eines Projekts

Es gibt verschiedene Möglichkeiten, bestimmte Teile eines Projekts zu compilieren. Sie können:

- einen einzelnen Knoten übersetzen;
- einen Knoten und seine Abhängigkeiten neu compilieren;
- einen Knoten und seine Abhängigkeiten aktualisieren;
- mehrere Knoten markieren und compilieren.

Um einen einzelnen Knoten zu übersetzen:

1. Markieren Sie den zu übersetzenden Knoten.
2. Wählen Sie *Projekt/Compilieren* aus dem Hauptmenü oder den Standard-Übersetzungsbefehl aus dem lokalen Menü. Wenn Sie beispielsweise eine .CPP-Datei markiert haben, enthält das lokale Menü des Knotens den Befehl *C++ Compilierung*, der nur den markierten Knoten compiliert.

Um einen Knoten und seine Abhängigen neu zu compilieren:

1. Markieren Sie den neu zu compilierenden Knoten.
2. Klicken sie ihn mit der rechten Maustaste an (oder drücken Sie [*Alt*][*F10*]) und wählen Sie *Knoten neu compilieren* aus dem lokalen Menü. Alle abhängigen Knoten werden ebenfalls compiliert, unabhängig davon, ob sie seit dem letzten Compilieren verändert wurden oder nicht.

Um einen Knoten und seine Abhängigen zu aktualisieren:

1. Markieren Sie den betreffenden Knoten.

2. Klicken sie ihn mit der rechten Maustaste an (oder drücken Sie [*Alt*][*F10*]) und wählen Sie *Knoten aktualisieren* aus dem lokalen Menü. Dieser Befehl compiliert nur die abhängigen Knoten, deren Quelldateien neuer als die zugehörigen Zieldateien sind.

Um mehrere Knoten zu compilieren:

1. Markieren Sie die betreffenden Knoten durch Drücken der Taste [*Strg*] und Anklicken (es müssen Knoten desselben Typs – z.B. .CPP – sein).
2. Wählen Sie zum Compilieren im lokalen Menü *Knoten neu compilieren* oder *Knoten aktualisieren*.

Beheben von Compilierzeit-Fehlern

Compilierzeit- oder *Syntaxfehler* treten auf, wenn Ihr Code eine Syntaxregel der Sprache verletzt, in der Sie programmieren. Der C++ Compiler kann Ihr Programm nicht compilieren, wenn es ungültige Ausdrücke enthält. Stößt der Compiler während des Compilierens auf einen Syntaxfehler, öffnet sich das Meldungsfenster und zeigt eine entsprechende Fehlermeldung oder Warnung. Über *Optionen/Umgebung/Vorgaben* können Sie festlegen, ob alte Meldungen zwischen den Aufrufen verschiedener Programmierwerkzeuge (z.B. Compiler, GREP oder Ressourcen-Compiler) erhalten bleiben oder gelöscht werden sollen. Markieren Sie *Alte Meldungen speichern*, wenn das Meldungsfenster die aktuelle Meldungsliste beim Start eines neuen Programmierwerkzeugs beibehalten soll.

Sie können das Meldungsfenster löschen, indem Sie in dessen lokalem Menü *Alle Meldungen entfernen* auswählen.

Fehler ansehen

Wenn Sie sich den Programmteil anzeigen lassen möchten, auf den sich eine Fehlermeldung oder Warnung bezieht, wählen sie die Meldung im Meldungsfenster aus. Die IDE liest daraufhin das Meldungsfenster neu aus, so daß in Ihrem Code die Stelle angezeigt wird, an der der Fehler oder die Warnung auftrat (dies wird als *Automatische Fehlersuche* bezeichnet). Ist die den Fehler enthaltende Datei nicht in einem Editierfenster geladen, so drücken Sie zum Laden [*Leer*] (Sie können die Datei auch durch Drücken von [*Alt*][*F10*] und Anklicken von *Quelltext öffnen* im lokalen Menü laden). Wenn Sie Fehler auf diese Weise ansehen, bleibt das Meldungsfenster geöffnet, so daß Sie die Meldungen nacheinander bearbeiten können. Um das Meldungsfenster zu öffnen oder anzusehen, klicken Sie auf die Schaltfläche im lokalen Menü oder wählen Sie *Ansehen/Meldung*.

Fehlerbehebung

Sie haben folgende Möglichkeiten, den Code, auf den sich eine Fehlermeldung oder Warnung bezieht, zu bearbeiten:

- Doppelklicken Sie auf die Meldung im Meldungsfenster.
- Markieren Sie die Meldung und drücken Sie [*Eingabe*].
- Drücken Sie [*Alt*][*F10*] und wählen Sie *Quelltext bearbeiten* aus dem lokalen Menü.

Der Cursor wird an die Stelle des Quelltextes bewegt, an der der Fehler verursacht wurde. Bearbeiten Sie Ihren Code von hier an, um den Fehler zu beheben. Mit der Tastenkombination [*Alt*][*F7*] springen Sie zur nächsten, mit [*Alt*][*F8*] zur vorherigen Fehlermeldung.

Compilieren in der IDE

Dieser Teil beschreibt, wie Sie einfache Programme mit Hilfe der Compiler-Optionen in der IDE compilieren. Wenn Sie wissen wollen, wie man große Projekte linkt, lesen Sie Kapitel 2.

Die IDE Mauspalette enthält drei Compiler-Buttons, die die folgenden Menü-Befehle ausführen:

Projekt/Compilieren [*Alt*]+[*F9*] compiliert die Datei im aktiven Editorfenster, wobei er die Compiler Optionen aus dem *Optionen/Projekt* Dialogfenster verwendet. Wenn ein Projekt-Fenster aktiv ist, werden alle ausgewählten Knoten des Projekts übersetzt. Kind-Knoten werden nicht übersetzt, wenn sie nicht auch ausgewählt sind. (siehe Kapitel 2 für weitere Informationen über Übersetzungen).

Projekt/Projekt aktualisieren [*F9*]übersetzt alle Knoten, die seit dem letzten Compilieren verändert wurden in einem Projekt. Wenn kein Projekt offen ist, werden alle Dateien in einem Editier- Fenster aufgebaut, indem der Projekt-Standard-Translator verwendet wird.

Projekt/Projekt neu compilieren übersetzt alle Knoten in einem Projekt - auch wenn sie seit dem letzten Compilieren nicht verändert wurden. *Projekt/Projekt aktualisieren* arbeitet das Projekt - beginnend mit dem ersten Knoten - sukzessive ab. Drücken Sie Verlassen, um das Linken zu beenden.

IDE Compiler Optionen

Es gibt zwei Möglichkeiten Optionen in der IDE zu setzen:

- Wählen Sie *Optionen/Projekt* aus und setzen Sie die Optionen in das Dialogfenster. Diese Optionen werden verwendet, wenn Sie ohne eine Pro-

jekt-Datei geladen zu haben compilieren. Wenn ein Projekt geladen ist, betrifft diese Option das ganze Projekt, das aufgebaut wird.

- Setzen Sie die Projekt-Optionen für jedes Projekt einzeln; Sie müssen dazu dem Projektmanager verwenden.Details dazu finden Sie in Kapitel 2.

Um z.B. Code in ein Editor-Fenster zu compilieren (Sie haben ein Projekt geladen), der eine 32-Bit-Anwendung für einen 80386 erzeugt,

1. Wählen Sie das Editierfenster mit dem zu compilierenden Code aus.
2. Wählen Sie *Optionen/Projekt* aus dem Menu aus.
3. In dem Projekt-Dialogfenster klicken Sie den 32-Bit Compiler an, um die Unterpunkte anzuzeigen, dann klicken Sie den Prozessor an.
4. Die Prozessor-Optionen erscheinen rechts. Markieren Sie 80386. Diese Option bleibt eingeschaltet, bis Sie sie wechseln oder die IDE verlassen. Um die Option als Standardoption zu speichern (so daß jedesmal, wenn Sie compilieren, Sie eine 32-Bit Anwendung für 80386 erhalten), wählen Sie *Optionen/Speichern* aus dem Hauptmenü.
5. Klicken Sie den Compilier-Button an oder wählen Sie Projekt/Compilieren (dieser Befehl compiliert den Code in das aktuelle Editorfenster, wenn kein Projekt geladen ist).

Kommandozeilen-Compiler

Dieser Teil beschreibt die Kommandozeilen Compiler (BCC.EXE und BCC32.EXE). BCC.EXE ist eine 16-Bit Anwendung, die einen 16-Bit Code erzeugt. BCC32.EXE ist eine 32-Bit Anwendung, die einen 32-Bit Code erzeugt. BCC und BCC32 arbeiten auf dieselbe Weise, aber haben verschiedene Standards (genauer erläutert in Tabelle 1.1) und erzeugen verschiedene Codes. Wenn nicht anders angegeben, gelten die Kommandos und Optionen für BCC auch für BCC32.

Sie können BCC benutzen, um Dateien zu TLINK oder TASM zu senden (ASM Dateien, wenn Sie TASM auf Ihrem Computer installiert haben). Die allgemeine Syntax für BCC.EXE lautet:

```
BCC [Option [Option..]] Dateiname [filename...]
```

Um eine Liste von allgemeinen Compileroptionen zu sehen, geben Sie BCC ohne jegliche Optionen oder Dateinamen ein und drücken Sie dann *ENTER*. Zwischen dem BCC-Kommando, jeder einzelnen Option und jedem einzelnen Dateinamen muß mindestens ein Leerzeichen stehen. Vor jeder Option steht entweder ein Bindestrich (-) oder ein Schrägstrich (/); um z.B. einen Include-Pfad Typen anzugeben:

```
BCC -Ic:\code\hfiles
```

Optionen und Dateinamen, die in die Kommandozeile eingegeben wurden, überschreiben die Einstellungen in der Konfigurationsdatei.

BCC compiliert Dateien ohne weitere Angaben und Datein mit .CPP standardmäßig als C++ Dateien. Dateien mit der Erweiterung .C und Angaben, die sich von den Erweiterungen .CPP, .OBJ, .LIB oder .ASM unterscheiden, werden als C Dateien compilert.

BCC.EXE und BCC32.EXE haben Optionen, die standardmäßig geöffnet sind (diese Optionen sind in der Tabelle 3.1 mit Punkten gekennzeichnet). Um eine Standardoption auszuschalten, geben Sie *BCC -option-* ein.

Standardmäßig versucht BCC, Dateien mit Moduldefinitionsdateien mit gleichnamigen EXE-Dateien zu linken. Benutzen Sie TLINK, um die Datei mit einer Moduldefinitionsdatei mit einem anderen Namen zu linken. Sie können nicht mit mehr als einer Moduldefinitionsdatei linken.

Konfigurationsdateien

Wenn Sie bestimmte Optionen immer wieder verwenden, können Sie diese in einer Konfigurationsdatei (als Standard ASCII Text Datei) zusammenfassen. Sie müssen die Optionen mit Leerzeichen trennen. Optionen können in einer oder mehreren Zeilen erscheinen.

TURBO.CFG konfiguriert BCC:EXE und BCC32.CFG konfiguriert BCC32.EXE; Projektdateien (.IDE) konfigurieren die IDE

BCC.EXE verwendet standardmäßig eine Konfigurationsdatei mit dem Namen TURBOC.CFG und entsprechend verwendet BCC32.EXE die Konfigurationsdatei BCC32.CFG (diese Standards sind in Tabelle 3.1 mit Punkten gekennzeichnet). Der Compiler sieht nach den .CFG Dateien erst in dem Verzeichnis nach, wo Sie BCC geschrieben haben und dann in dem Verzeichnis, in dem BCC.EXE und BCC32.EXE gespeichert sind.

Sie können mehrere Konfigurationsdateien anlegen oder TURBOC.CFG verändern.Um eine Konfigurationsdatei anzusprechen, schreiben Sie +*[Pfad]Dateiname* in die BCC Kommandozeile. Um z.B. eine Konfigurationsdatei mit dem Namen MYCONFIG.CFG zu verwenden, können Sie folgende Kommandozeile benutzen:

```
BCC +C:\MYCONFIG.CFG mycode.cpp
```

Optionen, die in der BCC Kommandozeile stehen, überschreiben Konfigurationsdateien.

Response-Dateien

Mit Hilfe von Response-Dateien können Sie Strings verwenden, die die normalerweise zugelassene Länge einer Kommandozeile überschreiten. Enthält die Kommandozeile sehr viele Optionen und/oder Dateien, können diese separat in eine ASCII-Textdatei geschrieben werden. Diese Datei nennen wir Response-Datei.

Um Response-Dateien zu verwenden,

1. Schreiben Sie die Kommandozeilen-Optionen, die Sie verwenden wollen, in eine Datei und speichern Sie diese Datei. Optionen können in einer oder mehreren Zeilen einer Datei stehen, sie müssen nur durch Leerzeichen getrennt sein. Response-Dateien von Borland C++ enden mit der Datei-Erweiterung RSP.
2. Schreiben Sie `BCC @[Pfad]respfile.rsp`.

Sie können mehr als eine Response-Datei anlegen, wenn Sie schreiben `BCC @[Pfad]respfile.rsp @[Pfad]otheresp.rsp`. Optionen, die in der Kommandozeile stehen, überschreiben jegliche Option oder jeglichen Dateinamen in einer Response-Datei.

Prioritätsregeln bei Optionen

BCC.EXE und BCC32.EXE bearbeiten die Optionen von links nach rechts und folgen den Regeln:

- Bei jeder duplizierten Option, außer bei **-I** und **-L**, überschreibt die zuletzt geschriebene Option jede vorherige.
- Wenn Sie mehrere **-I** oder **-L** Optionen in die Kommandozeile geschrieben haben, sucht der Compiler sich alle aufgelisteten Verzeichnisse von links nach rechts. Das Ergebnis ergibt sich aus der Kumulation der Optionen.

Optionen, die in der Kommandozeile eingegeben werden, überschreiben die Dateioptionen für Konfiguration und Antwort.

Compiler-Optionen-Tabelle

Compiler 3.1 enthält die Compileroptionen für die IDE und die Kommandozeile. Die meisten IDE Optionen erscheinen im *Optionen/Projekt* Dialogfenster. Einige Überschriften sind abgekürzt oder wiederholen sich in der Tabelle. Wenn sich eine Option nicht im Dialogfenster, dann erscheint die äquivalente Option bzw. das äquivalent Kommando der IDE in spitzen Klammern (**<>**). Sie finden eine detailliertere Erklärung der einzelnen Optionen auf den Seiten, auf die in der nachstehenden Tabelle verwiesen wird.

Standardoptionen für den 16-Bit und den 32-Bit Kommandozeilencompiler sind mit einem Punkt (■) markiert. Standardoptionen, die nur für den 16-Bit Kommandozeilencompiler gelten, sind mit diesem Punkt (■16) versehen und entsprechend ist der 32-Bit Compiler mit diesem Punkt (■32) versehen.

Es sei darauf hingewiesen, daß die Standards in der IDE verschieden sind. Der Hauptunterschied besteht darin, daß die IDE mit Debugger und Browserinformationen compiliert, wodurch Ihre compilerten Dateien

größer werden als wenn sie mit dem Kommandozeilencompiler compiliert worden wären (die Programme haben die gleiche Größe, wenn dieselben Optionen in beiden Anwendungen, der IDE und der Kommandozeile, verwendet worden sind).

Tabelle 3.1: Verzeichnis der Optionen

	Option	IDE-Menü	Beschreibung
	tdateiname«		Gibt dem Kommandozeilencompiler einen Kommandodateinamen an.
	+Dateiname	»keine	Veranlaßt den Kommandozeilencompiler, die alternative Konfigurationsdatei zu verwenden
	-1	»keine«	Generiert 80186 Befehle.
■16	-2	*16-Bit Compiler/Prozessor/80286*	Generiert 80286 kompatible Befehle für den Protected Mode (nur 16-Bit).
	-3	*16-Bit Compiler/Prozessor* 80386	Generiert 80386 kompatible Befehle für den Protected Mode (BCC Option).
■32	-3	*32-Bit Compiler/Prozessor* 80386	Generiert 80386 kompatible Befehle für den Protected Mode (BCC32 Option).
	-4	*16-Bit Compiler/Prozessor* 80486	Generiert 80486 kompatible Befehle für den Protected Mode (BCC Option).
	-4	*32-Bit Compiler/Prozessor* 80486	Generiert 80486 kompatible Befehle für den Protected Mode (BCC32 Option).
	-5	*32-Bit Compiler/Prozessor* Pentium	Generiert 32-Bit Pentium kompatible Befehle für den Protected Mode.
	-A	*Compiler/Quelltext/ANSI*	Ausschließliche Verwendung von ANSI-Schlüsselwörtern.
	-A-,-AT	*Compiler/Quelltext/Borland Erweiterungen*	Verwendung von Borland C++ Schlüsselwörtern.
	-AK	*Compiler/Quelltext/Kernighan und Ritchie*	Ausschließliche Verwendung von Kernighan und Ritchie Schlüsselwörtern.
	-AU	*Compiler/Quelltext/UNIX V*	Ausschließliche Verwendung von UNIX V Schlüsselwörtern.
	-an	*16-Bit oder 32-Bit Compiler/Prozessor/Byte, Wort, Doppel-Wort*	Ausrichtung nach: 1 = Byte, 2 = Word, 4 = Double Word (nur 32-Bit).
■	-a-	*16-Bit Compiler/Prozessor/Byte*	Ausrichtung auf ein Byte.
	-B	»keine«	Compiliert und ruft den Assembler zur Bearbeitung von asm-Anweisungen auf.
■	-b	*Compiler/Code-Generierung/Enum als Integer behandeln*	Aufzählungstypen haben Integergröße.
	-b-	*Compiler/Code-Generierung/Enum als Integer behandeln (Markierung löschen)*	*Aufzählungstypen haben (wenn möglich)* Byte-Größe.
	-C	*Compiler/Quelltext/Verschachtelte Kommentare*	Aktiviert die Verschachtelung von Kommentare.
■	-C-	*Compiler/Quelltext/Verschachtelte Kommentare (nicht markiert)*	Deaktiviert die Verschachtelung von Kommentaren.
	-c	*Projekt/Compilieren*	Compilierung zu .OBJ ohne Aufruf des Linkers
	-D*Name*	*Compileren/Definieren*	Definiert Name als einen Null-String.
	-D*Name=string*		Compileren/Definieren

	-d	*Compiler/Code-Generierung/Doppelte Strings zusammenfassen*	Doppelte Stringkonstanten nur einmal speichern.
■	-d-	*Compiler/Code-Generierung/Doppelte Strings zusammenfassen*	Doppelte Stringkonstanten getrennt speichern.
	-dc	*16-Bit Compiler/Speichermodell/Strings in Code-Segment speichern*	Verschieben Sie Ihre Stringkonstanten vom Datensegment in ein Codesegment (nur 16-Bit).
	-e*Dateiname*	»editiert Knoten-Attribute im Projektmanger«	
	-E*Dateiname*	»keine«	Verwendet den mit Dateiname bezeichneten Assembler.
	-Fc	*Compiler/Compilerausgabe/generiert*COMDEFs	Generiert COMDEFs (nur 16-Bit).
	-Ff	*16-Bit Compiler/Speichermodell/automatische Far-Daten*FAR-Daten	Legt automatisch FAR-Variablen an (nur 16-Bit).
	-Ff*size*	*16-Bit Compiler/Speichermodell/Schwellenwert für FAR-Daten*	Legt automatisch FAR-Variablen an; setzt Schwellengröße auf size (nur 16-Bit).
	-Fm	»keine«	Schaltet die Optionen -Fc,-Ff und -Fs ein (nur 16-Bit).
	-Fs	*TargetExpert/alternatives Startup*	Setzt DS=SS in allen Speichermodellen (nur 16-Bit DOS).
■	-f	*Compiler/Gleitkomma/keine Gleitkommaoperationen (nicht markiert)*	Emulieren von Gleitkommaoperationen.
	-f-	*Compiler/Gleitkomma/keine Gleitkommaoperationen*	Gleitkomma-Emulation aus.
■	-ff	*Compiler/Gleitkomma/schnelle Gleitkommaoperationen*	Schnelle Gleitkomma-Emulation.
	-ff-	*Compiler/Gleitkomma/schnelle Gleitkommaoperationen (Markierungen löschen)*	*Nur ANSI Gleitkomma-Emulation.*
	-f87	»keine«	Verwendung von 8087-Hardware-Instruktionen
	-f287	»TargetExpert und schnelle Gleitkommaoperationen«	Verwendung von 80287-Hardware Instruktionen (für DOS-Anwendungen).
	-G	*Optimierungen/spezielle Optimierungen/Geschwindigkeit*	Geschwindigkeitsoptimierung des erzeugten Codes.
■	-G-	*Optimierungen/spezielle Optimierungen/Größe*	Größenoptimierung des erzeugten Codes.
■	-gn	*Meldungen/Stoppen nach n Warnungen*	Warnungen: Stop nach n Meldungen (standardmäßig 100).
	-H	*Compiler/vorcompilierte Header/nicht erzeugen nicht verwenden*	Anweisung an den Compiler, vorcompilierte Header anzulegen und zu verwenden.
■	-H-	*Compiler/vorcompilierte Header/nicht anlegen und verwenden*	Ausschalten der Generierung und Verwendung von vorcompilierte Header.
	-Hc	»keine«	Speichern der vorcompilierten Header. Kann nur mit **-H** oder **-Hxxx** verwendet werden.
	-Hu	*Compiler/vorcompilierte Header/Benutzen, aber nicht anlegen*	Anweisung an den Compiler, vorcompilierte Header zu verwenden, aber keine anzulegen.
■	-H»xxxx«	*Compiler/vorcompilierte Header/Abbruch der Vorcompilierung nach der Header-Datei beenden*	*A*nweisung, das Compilieren von vorcompilierten Header bei der Datei (nur 32-Bit) zu beenden. Diese Option kann nur in Verbindung mit **-H**, **-Hu** oder **-H**=Dateiname verwendet werden.
	-H=Dateiname	*Compiler/vorcompilierte Header/Name der vorcompilierten Header-Datei*	Setzt den Dateinamen für vorcompilierte Header.

	-h	*16-Bit Compiler/Speichermodell/schnelle huge-Zeiger*	Verwendung der schnellen huge-Zeiger-Arithmetik (nur 16-Bit).
	-I*Suchpfad*	*Verzeichnisse/Include*	Suchpfad für Verzeichnisse mit Include-Dateien setzen.
■	-in	*Compiler/Quelltext/Bezeichnerlänge*	Signifikante Bezeichnerlänge auf n setzen (Standard ist 32).
■	-Jg	*C++ Optionen/Templates/automatisch*	Erzeugt Definitionen für alle Template-Instanzen; belegt bei Duplikaten nur einmal Speicherplatz.
	-Jgd	*C++ Optionen/Templates/global*	Erzeugt öffentliche Definitionen für alle Template-Instanzen; bei Duplikaten wird eine Fehlermeldung ausgegeben.
	-Jgx	*C++ Optionen/Templates/extern*	Erzeugt externe Referenzen auf alle Template-Instanzen.
■	-jn	*Compiler-Ausgabe/Stop nach n Fehlern*	Fehler: Stop nach n Meldungen
	-K	*Compiler/Code-Generierung/vorzeichenlose Zeichen*	voreingestellter Zeichentyp **unsigned**.
■	-K-	*Compiler/Code-Generierung/vorzeichenlose Zeichen (Markierung löschen)*	*voreingestellter Zeichentyp ist* **signed**.
	-K2	*C++-Optionen/C++-Kompatibilität*	Nur zwei Zeichen Typen (**unsigned** und **signed**) sind erlaubt. **char** wird als **signed** behandelt (nur 16-Bit). Kompatibilität mit Borland C++ 3.1 und früheren Versionen.
■	-k	*Compiler/Debugger/Standard Stack-Frame*	Voreingestellter Stack-Frame ein.
	-L*Suchpfad*	*Verzeichnisse/Bibliothek*	Suchpfad für Bibliotheksdateien setzen.
	-lx	*Linker/Optionen setzen*	Übergabe der Option x an den Linker (mehr als ein x möglich).
	-l-x	*Linker/Optionen setzen*	Option x für den Linker ausschalten
	-M	*Linker/.MAP-Datei/Segmente, Public oder Detailed*	weist den Linker an, eine .MAP-Datei zu erzeugen.
	-mc	*16-Bit Compiler, Speichermodell Compact*	Compilieren mit dem Speichermodell Compact (nur 16-Bit).
	-mh	*16-Bit 16-Bit Compiler/Speichermodell/Huge*	Compilieren mit dem Speichermodell Huge (nur 16-Bit).
	-ml	*16-Bit Compilierung/Speichermodell/Large*	Compilieren mit dem Speichermodell Large (nur 16-Bit).
	-mm	*16-Bit Compilierung/Speichermodell/Medium*	Compilieren mit dem Speichermodell Medium (nur 16-Bit).
	-mm!	*16-Bit Compilierung/Speichermodell/Medium und Never*	Compilieren mit dem Speichermodell Medium, wobei DS != SS (nur 16-Bit)
■	-ms	*16-Bit Compilierung/Speichermodell/Small*	Compilieren mit dem Speichermodell Small (nur 16-Bit).
	-ms!	*16-Bit Compilierung/Speichermodell/Small und Never*	Compilieren mit dem Speichermodell Small, wobei DS != SS (nur 16- Bit).
	-mt	*16-Bit Compilierung/Speichermodell/Tiny*	Compilieren mit dem Speichermodell Tiny (nur 16-Bit).
	-mt!	*16-Bit Compilierung/Speichermodell/Tiny und Nie*	Compilieren mit dem Speichermodell Tiny, wobei DS != SS (nur 16- Bit).
	-N	*Compiler/Debugger/Test auf Stack-Überlauf*	Überprüfen auf Stack-Überlauf
	-n*Suchpfad*	*Verzeichnisse/Ausgabeverzeichnisse*	Setzt das Ausgabeverzeichnis.
	-O	*Optimierungen/Größe/Sprungoptimierungen*	Optimiert Sprünge.

	-O1	*Optimierungen/Spezielle Optimierungen/Programmgröße*	Erzeugt den kleinstmöglichen Programmcode.
	-O2	*Optimierungen/Spezielle Optimierungen/Geschwindigkeit*	Erzeugt den schnellstmöglichen Programmcode (entspricht **-Ox**).
	-Oa	*Optimierungen/Spezielle Optimierungen/Kein Alias für Zeiger*	Optimierung für Zeigerausdrücke ohne Alias auf globale Unterausdrücke.
	-Ob	*Optimierungen/Größe/Entfernung überflüssiger Anweisungen*	Entfernt überflüssigen Programmcode.
	-Oc	*Optimierungen/Spezielle Optimierungen/Lokal Optimieren*	Entfernt überflüssiger Ausdrücke in Basisblöcken.
	-Od	*Optimierungen/Spezielle Optimierungen/Nicht optimieren*	Schaltet alle Optimierungen aus.
	-Oe	*Optimierungen/Größe/Globale Registerzuweisung*	Reserviert globale Register und analysiert die Lebensdauer der Variablen.
	-Og	*Optimierungen/Spezielle Optimierungen/Global Optimieren*	Entfernt redundante Ausdrücke innerhalb einer Funktion.
	-Oi	Optimierungen/Geschwindigkeit/*Inline-Erweiterung von intrinsic-Funktionen*	Aktiviert die Inline-Erweiterung von intrinsic-Funktionen.
	-Ol	*Optimierungen/Größe/Schleifenoptimierungen*	Verdichtet Schleifen.
	-Om	*Optimierungen/Geschwindigkeit/Verschiebung von invariantem Code*	Schiebt invarianten Code aus Schleifen.
	-Op	*Optimierungen/Geschwindigkeit/Kopien verwenden*	Verwendet Werte für Kopien von Konstanten, Variablen und Ausdrücken wieder, wo dies möglich ist.
	-Os	*Optimierungen/Spezielle Optimierungen/Programmgröße*	Erzeugt den kleinstmöglichen Code.
	-Ot	*Optimierungen/Spezielle Optimierungen/Programmgeschwindigkeit*	Erzeugt den schnellstmöglichen Programmcode (entspricht **-Oxt**).
	-Ov	*Optimierungen/Geschwindigkeit/Entfernnung von Induktionsvariablen*	Aktiviert oder desaktiviert die Erzeugung von Induktionsvariablen in Schleifen.
	-OW	*Optimierungen/Größe/Windows-Prolog/Epilog*	Unterdrückung von inc bp/dec bp für Windows FAR-Funktionen (Nur 16-Bit).
	-Ox	*Optimierungen/Spezielle Optimierungen/Programmgeschwindigkeit*	Erzeugt den schnellsten Programmcode; Kompatibel zu Microsoft.
	-o*Dateiname*	»keine«	Kompilieren der Quelldatei nach *Dateiname*.OBJ.
■	-P	benutze Tools	C++-Compilierung ausführen, unabhängig von der Namenserweiterung der Quelltextdatei.
	-P*ext*	»benutze Tools«	C++-Compilierung ausführen und *ext* als Standard-Namenserweiterung vergeben.
■	-P-	»benutze Tools«	C++- oder C-Compilierung ausführen und die Namenserweiterung abhängig von der der Quelltextdatei vergeben (Voreinstellung).
	-P-*ext*	»benutze Tools«	C++- oder C-Compilierung in Abhängigkeit von der Namenserweiterung ausführen; die Standard-Namenserweiterung auf *ext* setzen.
	-p	*16-Bit Compiler/Aufrufkonvention/Pascal*	Verwendung der Pascal-Aufrufkonvention für 16-Bit-Anwendungen (BCC-Option).
	-p	*32-Bit Compiler/Aufrufkonvention/Pascal*	Verwendung der Pascal-Aufrufkonvention für 32-Bit-Anwendungen (BCC32-Option).
■	-p- -pc	*16-Bit Compiler/Aufrufkonvention/C*	Verwendung der Aufrufkonvention von C (BCC-Option).

■	-p- -pc	*32-Bit Compiler/Aufrufkonvention/C*	Verwendung der Aufrufkonvention von C (BCC32-Option).
	-po	*Compiler/Code-Generierung/FastThis*	Verwendung der **fastthis**-Aufrufkonvention für die Übergabe von 'this'-Parametern in Register (nur 16-Bit).
	-pr	*16-Bit Compiler/Aufrufkonvention/Register*	Verwendung der fastcall-Aufrufkonvention für die Übergabe von Parametern in Register (BCC-Option).
	-pr	*32-Bit Compiler/Aufrufkonvention/Register*	Verwendung der fastcall-Aufrufkonvention für die Übergabe von Parametern in Register (BCC32-Option).
	-ps	*32-Bit Compiler/Aufrufkonvention/Automatisch*	Verwendung der stdcall-Aufrufkonvention (nur 32-Bit).
■	-r	*Compiler/Code-Generierung/Automatisch*	Verwendung von Register-Variablen.
	-r-	*Compiler/Code-Generierung/Keine*	Keine Verwendung von Register-Variablen.
	-rd	*Compiler/Code-Generierung/Register-Schlüsselwort*	Nur deklarierte Register-Variablen in Registern erlaubt.
	-R	*Compiler/Fehlersuche/ObjectBrowser-Informationen*	Einfügen von Browser-Informationen in generierte .OBJ Dateien.OBJ-Dateien.
■	-RT	*C++-Optionen/Ausnahmebehandlung/RTTI/Aktivierung von RTTI*	Aktiviert die Laufzeit-Typinformation (Run-time type information).
	-S	Projekt Manager Geschwindigkeits-Menü	Erstellen einer .ASM-Ausgabedatei.
	-T*string*	»benutze Tools«	Übergabe von *string* als Parameter an TASM, TASM32 oder Assembler, mit **-E** definiert.
	-T-	»Benutze Werkzeuge«	Löschen aller vorausgehenden Assembler-Parameter.
	-tD	»TargetExpert«	Legt das Ziel als DOS .EXE-Datei fest.
	-tDc	»TargetExpert«	Legt das Ziel als DOS .COM-Datei fest.
■	-tDe	*TargetExpert*Legt das Ziel als DOS .EXE-Datei fest.	
	-tW	*16-Bit Compiler/Entry/Exit-Code/Windows: alle Funktionen exportierbar*	Legt das Ziel als Windows .EXE-Datei fest; alle Funktionen sind exportierbar.
	-tWC	*TargetExpert*	Legt das Ziel als Konsole .EXE-Datei fest; alle Funktionen sind exportierbar.
	-tWCD	*»TargetExpert«*	Legt das Ziel als Konsole .DLL-Datei fest; alle Funktionen sind exportierbar.
	-tWCDE	»TargetExpert«	Legt das Ziel als Konsole .DLL-Datei fest; ausgewählte Funktionen sind exportierbar.
	-tWD	*16-Bit Compiler/Entry/Exit-Code/Windows-DLL: alle Funktionen exportierbar*	Legt das Ziel als Windows .DLL-Datei fest; alle Funktionen sind exportierbar.
	-tWDE	*16-Bit Compiler/Entry/Exit-Code/Windows-DLL: ausgewählte Funktionen exportierbar*	Legt das Ziel als Windows .DLL-Datei fest; ausgewählte Funktionen sind exportierbar.
	-tWE	*16-Bit Compiler/Entry/Exit-Code/Windows: explizite Funktionen exportieren*	Legt das Ziel als Windows .EXE-Datei fest; ausgewählte Funktionen sind exportierbar.
	-tWM	»TargetExpert und Multi-Thread prüfen«	Erzeugt ein Multi-Thread Ziel (Nur 32-Bit).
	-tWS	*16-Bit Compiler/Entry/Exit-Code/Windows intell. Callbacks, alle Funktionen exportierbar*	Legt das Ziel als Windows .EXE-Datei fest, die Smart Callbacks verwendet

			(nur 16-Bit); alle Funktionen sind exportierbar.
	-tWSE	*16-Bit Compiler/Entry/Exit-Code/Windows: intell. Callbacks, ausgewählte Funktionen exportierbar*	*Legt das Ziel als Windows .EXE-Datei* fest, die Smart Callbacks verwendet (nur 16-Bit); ausgewählte Funktionen sind exportierbar.
	-U*Name*	»Benutze lokales Überschreiben in Projekten«	Alle vorausgehenden *Name*-Definitionen werden aufgehoben.
■	-u	*Compiler/Compiler-Ausgabe/Unterstriche erzeugen*	Generieren von Unterstrichen ein (Voreinstellung).
	-v, -v-	*Compiler/Debugger/Debug-Informationen in OBJ-Dateien*	Quelltext-Debugging ein-/ausschalten.
	-vi, -vi-	*Compiler/Fehlersuche/Out-of-line Inline-Funktionen*	Steuert die Erweiterung von Inline-Funktionen.
	-V0	*C++-Optionen/Virtuelle Tabellen/Extern*	Externe virtuelle C++-Tabellen.
	-V1	*C++-Optionen/Virtuelle Tabellen/Öffentlich*	Öffentliche virtuelle C++-Tabellen.
■	-V	*C++-Optionen/Virtuelle Tabellen/lokal*	Verwendung von lokalen virtuellen C++-Tabellen.
	-Va	*C++-Optionen/C++ Kompatibilität/Klassenwerte als Referenz übergeben*	Übergabe von Klassenargumenten durch Referenz an eine temporäre Variable.
■	-Vb	*C++-Optionen/C++-Kompatibilität/Gleiche Größe wie 'this'-Zeiger*	Bewirkt, daß der Basisklassen-Zeiger so groß ist wie der 'this'-Zeiger der Klasse (Nur 16-Bit).
■16	-Vb-	C++-Optionen/C++-Kompatibilität/Immer near	Macht den virtuellen Basisklassen-Zeiger immer **near** (nur 16-Bit).
	-Vc	*C++-Optionen/C++-Kompatibilität/Konstruktorersetzungen abschalten*	Fügt verborgene Elemente und verborgenen Programmtext nicht zu Klassen hinzu, deren Zeiger auf virtuelle Basisklassen-Elemente zeigen (nur 16-Bit).
	-Vf	*Compiler/Speichermodell/Virtuelle Tabellen als FAR*	FAR C++ virtuelle Tabellen (nur 16-Bit).
	-Vmd	*C++-Optionen/Elementzeiger/Kleinstmöglicher für Klasse*	Verwendung der kleinsten Darstellung von Elementzeigern.
	-Vmm	*C++-Optionen/Elementzeiger/Unterstützung von Mehrfachvererbung*	Elementzeiger-Unterstützung von Mehrfachvererbung.
	-Vmp	*C++-Optionen/Elementzeiger/Präzision von Elementzeigern berücksichtigen*	Berücksichtigung der deklarierten Genauigkeit für alle Elementzeiger-Typen.
	-Vms	*C++-Optionen/Elementzeiger/Unterstützung von Einfachvererbung*	Elementzeiger-Unterstützung von Einfachvererbung.
■	-Vmv	C++-*Optionen/Elementzeiger/Unterstützung aller Fälle*	Elementzeiger ohne Einschränkungen (allgemeinste Repräsentation).
	-Vo	»keine«	Aktiviert die aller *-V*-Optionen (-Va, -Vb, -Vc, -Vp, -Vt, -Vv).
	-Vp	*C++-Optionen/C++ Kompatibilität/echte 'Pascal'-Elementfunktionen*	Übergibt 'this'-Parameter an **pascal**-Elementfunktionen als ersten Parameter für den Stack.
	-Vs	C*++-Optionen/Virtuelle Tabellen/lokal*	Lokale virtuelle C++ Tabellen.
	-Vt	*C++-Optionen/C++ Kompatibilität/Zeiger auf virtuelle* Tabelle hinter den Datenelementen	Plaziert den virtuellen Tabellenzeiger nach nicht-statischen Datenelementen.
	-Vv	*C++-Optionen/C++ Kompatibililtät/'Tiefe' virtuelle Basisklassen*	Fügen Sie keine verborgenen Elemente hinzu und codieren Sie nicht nach Klassen mit Zeigern auf Klassenelemente einer virtuellen Basis.

	-W	*16-Bit Compiler/Entry/Exit-Code/Windows: Alle Funktionen exportieren*	Erzeugen einer .EXE Datei für Windows mit Funktionen, die alle für den Export bestimmt sind.
	-WD	*16-Bit Compiler/Entry/Exit-Code/Windows-DLL, Alle Funktionen exportierbar*	Erzeugen einer .DLL Datei für Windows mit Funktionen, die alle für den Export bestimmt sind.
	-WDE	*16-Bit Compiler/Entry/Exit-Code/Windows-DLL:*	Erzeugen einer .DLL Datei für Windows mit Funktionen, die explizit für den Export bestimmt sind.
	-WE	*16-Bit Compiler/Entry/Exit-Code/Windows: Explizite Funktionen exportieren*	Erzeugen einer .EXE Datei für Windows mit Funktionen, die explizit für den Export bestimmt sind.
	-WM	»TargerExpert und multithread überprüfen«	Erzeugen einer .EXE Datei für Multi-Thread Windows mit Funktionen, die explizit für den Export bestimmt sind (Nur 32-Bit).
	-WS	*16-Bit Compiler/Entry/Exit/Windows intell. Callbacks, alle Funktionen exportierbar*	Erzeugen einer .EXE Datei für Windows, die Smart Callbacks verwendet, mit Funktionen, die für den Export bestimmt sind (nur 16-Bit).
	-WSE	*16-Bit Compiler/Entry/Exit/Windows intell. Callbacks, explizite Funkt. exportierbar*	Erzeugen einer .EXE Datei für Windows, die Smart Callbacks verwendet, mit Funktionen, die explizit für den Export bestimmt sind (nur 16-Bit).
	-w!	*Meldungen/Abbruch nach x Warnungen*	Gibt einen Wert ungleich Null von dem Komandozeilencompiler zurück, wenn x Warnungen aufgetreten sind und compiliert nicht zu einer .OBJ.
■	-w	*Meldungen/Alle*	Anzeige von Warnungen.
	-w-	*Meldungen/Keine*	Zeigt die Warnungen nicht an.
	-w»xxx«	*Meldungen/Ausgewählte (siehe spezielle Warnungen)*	Aktiviert Warnung xxx.
	-w-»xxx«	*Meldungen/Ausgewählte (siehe spezielle Warnungen)*	Deaktiviert Warnung xxx.
	-X	Compiler/Compiler Output/Abhängigkeitsausgaben (nicht markiert)	Schaltet Compiler-Abhängigkeitsausgaben aus.
■	-X-	*Compiler/Compiler Output/Informationen über automatische Abhängigkeiten*	Schaltet Compiler-Abhängigkeitsausgaben ein.
■	-x	*C++-Optionen/Exceptions/RTTI/Ausnahmebehandlung aktivieren*	Aktivieren der Ausnahmebehandlung.
■	-xd	*C++-Optionen/Exceptions/RTTI/Destruktoraufrufe einschalten*	Aktivieren des Destruktor Cleanup.
	-xp	*C++-Optionen/Exceptions/RTTI/Positionsinformationen einschalten*	Aktivieren des Erzeugung von Informationen zum Auffinden von Ausnahmen.
	-Y	»Target Expert DOS Overlay«	Ermöglicht Overlay-Codegenerierung.
	-Yo	*»ditiert die Knotenattribute und kontrolliert das Overlay«*	Compiliert die Dateien als Overlays.
	-y	*Compiler/Debugger/Zeilennummern*	Zeilennummern ein.
	-Z	*Optimierung/Größe/Globale Registerzuweisung*	Optimierter Gebrauch von Registern.
	-zA*name*	*16-Bit Compiler/Code-Segmentnamen/Klasse*	Code-Klasse.
	-zB*name*	*16-Bit Compiler/Datensegmentnamen/Uninitialisierte Daten*	BSS-Klasse.
	-zC*name*	*16-Bit Compiler/Code-Segmentnamen/Code*	Code-Segment.
	-zD*name*	*16-Bit Compiler/Datensegmentnamen/Uninitialisierte Daten*	BSS-Segment (Nur 16-Bit).
	-zE*name*	*16-Bit Compiler/Far-Datensegmentnamen/Segment*	FAR-Segment (nur 16-Bit).
	-zF*name*	*16-Bit Compiler/Far-Datnsegmentnamen/Klasse*	FAR-Klasse (nur 16-Bit).
	-zG*name*	*16-Bit Compiler/Datensegmentnamen/Uninitialisierte Daten*	BSS-Gruppe.

	-zH*name*	*16-Bit Compiler/FAR-Datensegmentnamen/Far Daten*	FAR-Gruppe (nur 16-Bit).
	-zP*name*	*16-Bit Compiler/Code-Segmentnamen/Gruppe*	Code-Gruppe.
	-zR*name*	*16-Bit Compiler/Datensegmentnamen/Segment*	Datensegment.
	-zS*name*	*16-Bit Compiler/Datensegmentnamen/Initialisierte Daten*	Datengruppe.
	-zT*name*	*16-Bit Compiler/Datensegmentnamen/Initialisierte Daten Klasse*	Datenklasse.
	-zV*name*	*16-Bit Compiler/FAR-Datensegmentnamen/Virtuelle FAR-Tabellen Segment*	Virtuelles FAR-Segment (nur 16-Bit).
	-zW*name*	*16-Bit Compiler/FAR-Datensegmentname/Virtuelle FAR-Tabellen Klasse*	virtuelle FAR-Klasse (nur 16-Bit).
■	-zX*	»keine«	Verwendung der Standardnamen für *X*; *X* ist A-H, P, R, S, T, V oder W.

Zeichenerklärung:

■ Vorgabe für 16- und 32-Bit

■16 Vorgabe für 16-Bit

■32 Vorgabe für 32-Bit

Verzeichnisse

-I*Pfad* **Include** sucht in *Pfad* (Laufwerkbezeichnung oder Pfadnamen eines Unterverzeichnisses) nach den einzubindenden Dateien (zusätzlich zum Durchsuchen der Standardverzeichnisse).Die Laufwerkbezeichnung besteht aus einem einzelnen Klein- oder Großbuchstaben, der von einem Doppelpunkt (:) gefolgt wird. Unterverzeichnis kann jedes Verzeichnis und jeder gültige Verzeichnispfad sein. Die Verzeichnisoption **-I** (Großbuchstabe I) kann mehrfach verwandt werden.

-L*Pfad* **Library** läßt den Linker die C0x.OBJ start-up Objektdatei und die Borland C++ Bibliotheksdateien im angegebenen Verzeichnis suchen. Standardmäßig sucht der Linker im aktuellen Verzeichnis.

Quelltext ist das Verzeichnis, in dem der Compiler den Quelltext sucht.

Obj-Dateien ist das Verzeichnis, in dem der Compiler alle temporären Dateien ablegt, die er erzeugt.

n*Pfad* **Exe-Dateien** legt jede vom Compiler erzeugte Outputdatei (.OBJ, .I oder .ASM) in das Verzeichnis oder Laufwerk, das durch *Pfad* angegeben ist.

- Sie können in der Kommandozeile verschiedene Verzeichnisse auf folgende Art und Weise eingeben:
- Sie können verschiedene Einträge mit einer einzelnen **-L** oder **-I** Option stapeln, wenn Sie das Semikolon verwenden:

```
BCC.EXE –LVerzeichnisname1;Verzeichnisname2;Verzeichnisname3
–Iinc1;inc2;inc3 Dateiname.c
```

- Sie können folgendermaßen mehr als eine Option in der Kommandozeile verwenden:

```
BCC.EXE -LVerzeichnisname1 -LVerzeichnisname2 -LVerzeichnisname3 -Iinc1
-Iinc2 -Iinc3 Dateiname.c
```

- Sie können Mischformen verwenden:

```
BCC.EXE -LVerzeichnisname1;Verzeichnisname2 -LVerzeichnisname3
-Iinc1;inc2 -Iinc3 Dateiname.c
```

Wenn Sie verschiedene **-L** oder **-I** Optionen in der Kommandozeile verwenden, ist das Ergebnis kumulativ: Der Compiler sucht alle aufgelisteten Verzeichnisse von links nach rechts.

Die IDE unterstützt verschiedene Bibliotheksverzeichnisse.

Datei-Such Algorithmus

Der Borland C++-Suchalgorithmus für Includedateien sucht die Header, die in Ihrem Quelltext aufgelistet sind, auf folgende Weise:

- Wenn Ihr Quelltext ein #include `<somefile.h>` enthält, sucht Borland C++ nach dem somefile.h nur in dem speziellen Includeverzeichnis.
- Wenn Ihr Quelltext ein #include »somefile.h« enthält, sucht Borland C++ erst in dem aktuellen Verzeichnis; Kann es die Datei nicht finden, sucht C++ in dem in der Kommandozeile angegebenen Verzeichnis.

Der Bibliothek-Suchalgorithmus ist dem der Includedateien sehr ähnlich:

- Implizite Bibliotheken: Borland C++ such nach impliziten Bibliotheken nur in den speziellen Bibliotheksverzeichnissen; dies ist dem Suchalgorithmus für `+include <somefile.h>` sehr ähnlich. Implicite Bibliotheksdateien werden von Borland C++ automatisch in die start-up Objektdatein (C0x.OBJ) gelinkt.
- Explizite Bibliotheken: Wo Borland C++ nach expliziten (benutzerspezifischen) Bibliotheken sucht, hängt zum Teil davon ab, wie die Bibliotheksdateinamen aufgelistet sind. Explizite Bibliotheken werden entweder in der Kommandozeile oder in einer Projektdatei aufgelistet. Es handelt sich dabei um die Dateien mit der Namenerweiterung .LIB.
 - Wenn Sie einen expliziten Bibliotheksdateinamen ohne Laufwerk oder Verzeichnis (z.B.:mylib.lib) auflisten, sucht Borland C++ diese Bibliothek erst in dem aktuellen Verzeichnis. Erst dann (wenn die erste Suche erfolglos war) sucht C++ in den speziellen Bibliotheksverzeichnissen. Dies ist ähnlich wie bei der Suche nach **-include** *»somefile.h«*.
 - Wenn Sie eine benutzerspezifische Bibliothek mit Laufwerk und/oder Verzeichnisinformation (z.B.:c:mystuff\mylib1.lib) angeben, sucht Borland C++ *nur* in dem angegebenen Laufwerk und/oder Verzeichnis.

Compiler definiert

Mit Hilfe von Makro-Definitionen können Sie in der Kommandozeile oder in der IDE Makros definieren oder deren Definition aufheben (sog. manifeste oder symbolische Konstanten). In der Kommandozeile oder in Dialogfenstern wie *Optionen/Setzen* definierte Makros überschreiben die entsprechenden Makros in der Quelltextdatei.Schreiben Sie die Makrodefinitionen in das Dialogfenster Definitionen unter den Menüpunkt Code-Erzeugung/Einstellungen.

-Dname Definiert den Bezeichner *Name* als leeren String. Trennen Sie Makros mit Semikolons.

-Dname = string Definiert den Bezeichner *Name* als den String *String*, der ihm nach dem Gleichheitszeichen zugewiesen wird. *String* darf weder Leerzeichen noch Tabulatoren enthalten. Makros werden durch Semikolons getrennt.

Borland C++ ermöglicht auf folgende Art und Weise die Eingabe mehrerer **#define**s in die Kommandozeile:

- Sie können nach einer einzigen -D-Option mehrere Angaben machen, die durch das Semikolon voneinander getrennt sind:

```
BCC.EXE -Dxxx;yyy=1;zzz=NO MYFILE.C
```

- Sie können mehrere -D-Optionen in der Kommandozeile eingeben, die durch Leerzeichen getrennt werden:

```
BCC.EXE -Dxxx -Dyyy=1 -Dzzz=NO MYFILE.C
```

- Sie können beide Möglichkeiten kombinieren:

```
BCC.EXE -Dxxx -Dyyy=1;zzz=NO MYFILE.C
```

Compiler/Code-Erzeugung

-b -b- sorgt dafür, daß der Compiler immer ein 2-byte **int** (16-Bit) oder vier-byte **int** (32-Bit) für Aufzählungstypen bereithält. Diese Option ist standardmäßig eingeschaltet. Ist die Option ausgeschaltet (**-b-**), wird dafür gesorgt, daß die kleinste Variablengröße, die den Aufzählungstypen aufnehmen kann, bereitgestellt wird: Der Compiler wird veranlaßt, ein **unsigned** oder **signed char** zu reservieren, wenn der Minimal- und Maximalwert der Aufzählungstypen innerhalb des Bereiches 0 bis 255 bzw. -128 bis 127 liegt, oder einen **unsigned** oder **signed short** zu reservieren, wenn der Minimal- und Maximalwert innerhalb des Bereiches 0 bis 65,535 oder -32,768 bis 32,767 liegt. Der Compiler stellt ein zwei-byte **int** (16-Bit) oder ein vier-byte **int** (32-Bit) für die Aufzählungstypen bereit, wenn irgendein Wert außerhalb dieser Bereiche liegt.

-K -K- behandelt alle **char**-Defintionen als **unsigned characters**, auch wenn sie vom Typ signed sind. Dies stellt die Kompatibilität zu Compilern her, die char-Deklarationen als unsigned behandeln. Die BCC Voreinstellung für **char**-Deklarationen ist **signed** (nicht-markiert oder -K-).

-d weist den Compiler an, denselben Speicherplatz zu verwenden, wenn zwei Strings identisch sind. Das Programm wird dadurch kürzer, die Zeit zum Compilieren wird etwas länger. Diese Option kann Fehler hervorrufen, wenn ein String verändert wird. Diese Option ist standardmäßig ausgeschaltet (**-d-**).

po **fastthis** verwendet den **__fastthis** Aufruf, um **this** in ein Register der Elementfunktionen zu übergeben.

fastthis ermöglicht (nur bei 16-Bit Anwendungen, da **fastthis** für 32-Bit Anwendungen immer verwendet wird), daß der Compiler die Elementfunktionen compiliert, um deren 'this'-Zeiger in ein Register zu übergeben (oder ein Registerpaar in einem 16-Bit Datenmodell). Gleichermaßen laden Elementfunktionsaufrufe das Register (oder Registerpaar) mit 'this'.

Sie können **fastthis** aufrufen, indem sie die **-po** Kommandozeilenoption oder die Compiler/Code-Generierung/FastThis Aufrufoption verwenden. Sie können auch die Sprachanweisung **__fastthis** verwenden.

In kleinen oder flachen Datenmodellen ist 'this' in dem SI-Register eingefügt; 16-Bit große Datenmodelle verwenden DS:SI. Wenn es notwendig ist, speichert der Compiler DS und holt DS zurück. Alle Verweise in einer Elementfunktion auf Elementdaten werden über das SI-Register gegeben.

Die Namen von Elementfunktionen, die mit fastthis compiliert worden sind, werden verschieden ergänzt von non-fastthis Elementfunktionen, um ein Vermischen der beiden zu verhindern. Am leichtesten ist es, wenn Sie alle Klassen mit fastthis compilieren, aber Sie können auch einige Klassen mit fastthis compilieren und andere nicht.

-r- **None** benutzt keine Registervariablen.

-rd **Register keyword** gibt an, daß Registervariablen nur verwendet werden, wenn Sie das Register Schlüsselword verwenden und ein Register zugreifbar ist. Sie können **-rd** in **#pragma option**s verwenden. Verwenden Sie diese Option oder die **-r**-Option, um die Registerverwendung zu optimieren.

-r Verwendet **automatisch** Registervariablen. Der Compiler weist Registern, wenn möglich, automatisch Variablen zu, indem er den Register Typ Spezifizierer verwendet, auch wenn Sie keine Registervariable spezifizieren. Die **-r**-Option ist voreingestellt.

Compiler/Gleitkomma

-f -f- **Kein Gleitkomma** (-f-) teilt Borland C++ mit, daß das Programm keine Gleitkomma-Berechnungen enthält, so daß keine Gleitkommabibliotheken gelinkt werden müssen. Ist die Option nicht markiert (-f), ist der Compiler angewiesen, 80x87-Aufrufe zur Laufzeit zu emulieren.

-ff -ff- **Schnelle Gleitkommaberechnung** (-ff) optimiert die Gleitkommaoperation, ohne auf explizite oder implizite Typumwandlung zu achten. Berechnungen erfolgen schneller, als unter dem ANSI-Modus. Ist diese Option ausgeschaltet (**-ff-**), schaltet die Option die schnelle Gleitkommaroutinen aus. Bei der Compilierung werden strikt die ANSI-Regeln hinsichtlich der Gleitkomma-Umwandlung befolgt.

Compiler/Compilerausgabe

-X -X- **Informationen über die automatische Abhängigkeitsprüfung** (**-X**) erzeugt Informationen über die automatische Abhängigkeitsprüfung. Module, die mit dieser Option compiliert werden, können die automatische Abhängigkeitsprüfung von Make verwenden. Standardmäßig ist die automatische Abhängigkeitsprüfung eingeschaltet (-X-).

-u die **Unterstricherzeugung** wird von dem Compiler gemacht. Der Compiler wird angewiesen, jedem Bezeichner vor dem Abspeichern im Modul automatisch einen Unterstrich voranzustellen. Unterstriche für C und C++ Bezeichner sind optional, sind aber voreingestellt. Sie können dies mit -u- ausschalten. Wenn Sie jedoch mit den Standardbibliotheken von C++ arbeiten, kann es zu Problemen führen.

-Fc veranlaßt die **Generierung von gemeinsamen Variablen** (COMDEFs) für globale C-Variablen, die nicht initialisiert sind und nicht als **static** oder **extern** deklariert sind. Der Vorteil dieser Option liegt darin, daß die in verschiedenen Quelltextdateien vorhandenen Header Daklarationen globaler Variablen enthalten können. Solange eine Variable nicht mit einem anderen Wert als Null initialisiert werden muß, muß sie auch in den Quelltextdateien nicht definiert werden. Sie können diese Option einsetzen, wenn Sie Code für eine andere Implementierung mit ähnlichen Eigenschaften portieren wollen.

Compiler/Quelltext

-C erlaubt die **Verschachtelung von Kommentaren**. Kommentare dürfen normalerweise nicht verschachtelt sein.

-in **Bezeichnerlänge**: Die **Bezeichnerlänge** ist für den Compiler insofern wichtig, als nur die ersten *n* Zeichen eines Bezeichners zu berücksichtigen sind. Alle Bezeichner, wie Variablen, Makronamen von Präprozessoren und Namen von Strukturelementen werden nur dann als unterschiedlich angesehen, wenn sich ihre ersten *n* Zeichen voneinander unterscheiden. Wenn *n* als null oder größer als 249 angegeben ist, oder wenn die **-in**-Option nicht eingeschaltet ist, sind auch Bezeichner mit unbeschränkter Länge erlaubt.

Per Voreinstellung zieht Borland C++ 32 Zeichen pro Bezeichner heran. Andere Systeme, einschließlich einiger Unix-Compiler, berücksichtigen nur die ersten 8 Zeichen. Wenn Sie Code in solche Umgebung portieren, können Sie den Code mit einer niedrigeren Anzahl von berücksichtigten Zeichen compilieren. Eine derartige Compilierung hilft bei der Aufdeckung von Namenskonflikten bei langen Bezeichnern, wenn die Anzahl der berücksichtigten Zeichen herabgesetzt wurde.

-A- -AT **Borland Erweiterungen** erlauben die Verwendung von C++ Schlüsselwörtern.

-A **ANSI** compiliert ANSI-kompatiblen Code. Alle in Borland C++ reservierten Wörter verlieren ihre spezielle Bedeutung und können als normale Bezeichner verwendet werden.

-AU **UNIX V** benutzt ausschließlich Unix-Schlüsselwörter.

-AK **Kernighan and Richie** benutzt ausschließlich Kernighan und Richie-Schlüsselwörter.

Compiler/Debugger

-k **Standard-Stack-Frame** erzeugt einen Standard-Stack-Frame, der beim Debuggen sehr nützlich ist, um den Stack der Unterprogrammaufrufe zu überprüfen. Diese Option ist in der Voreinstellung. Wenn diese Option ausgeschaltet ist, wird jede Funktion, die keine lokalen Variablen verwendet und keine Parameter besitzt, mit abgekürztem Eingabe- und Rückgabewert compiliert, um den Code kleiner und schneller zu machen.

-N **Stack-Überlauf-Test** erzeugt bei jedem Funktionseinsprung einen Code zur Erkennung von Stack-Überlauf, der eine Meldung ausgibt, sobald ein Überlauf während der Laufzeit entdeckt wird.Das Programm wird dadurch größer und langsamer, aber die Option ist eine Möglichkeit, die nur schwer entdeckbaren Stach-Überläufe zu finden.Bei einem Stack-Überlauf erhalten Sie die Meldung Stack overflow, danach wird das Programm mit dem Exit-Code 1 abgebrochen.

-vi **Inline-Funktionen** erweitern C++ inline-Funktionen inline.Um die Erweiterung von inline-Funktionen zu kontrollieren, verhält sich die **-v**-Option

etwas anders in C++: Ist die Erweiterung der inline-Funktion ausgeschaltet, wird die Funktion wie jede andere Funktion generiert und aufgerufen. Da sich die Fehlersuche bei eingeschalteter inline-Funktion äußerst schwierig gestalten kann, verfügt Borland C++ über folgende Optionen:

- **-v** schaltet die Dabugger-Information ein und die **inline**-Funktion aus. Wenn diese Option ausgeschaltet ist, können Sie größere .OBJ Dateien linken. Diese Option wirkt sich nicht auf die Ausführungsgeschwindigkeit aus, aber auf die Compilierzeit.
- **-v-** schaltet die Debugger-Information und die **inline**-Funktion aus.
- **-vi** schaltet die **inline**-Erweiterung ein.
- **-vi-**schaltet die **inline**-Funktion aus.

Zum Beispiel müßten zum Einschalten der Fehlersuche und der inline-Erweiterung demnach die beiden folgenden Optionen verwendet werden: **-v -vi**.

-y **Zeilennummern** schließt Zeilennummern in der .OBJ für den in der IDE integrierten Debugger ein. Dadurch wird zwar die .OBJ-Datei größer, die Ausführungsgeschwindigkeit des ausführbaren Programms bleibt jedoch unverändert. Diese Option ist sinnvoll in Zusammenhang mit einem symbolischen Debugger. Generell läßt sich sagen, daß der Gebrauch von **-v** für den integrierten Debugger besser ist als die Verwendung von **-y**. Die Debugger Optionen schließen Debuggerinformationen in Ihrem erzeugten Code ein.

-v Debugger-Informationen in den OBJs schließen Debugger-Informationen in den .OBJ-Dateien ein. Der Compiler wird angewiesen, Debugger-Informationen in die .OBJ-Datei aufzunehmen, damit die Fehlersuche sowohl mit dem integrierten Debugger als auch mit dem Standalone-Debugger durchgeführt werden kann. Der Compiler übergibt diese Funktion auch an den Linker, damit die Debugger-Information auch in der .EXE-Datei enthalten ist. Zur Fehlersuche sorgt diese Option außerdem dafür, daß C++ inline-Funktionen als normale Funktionen gehandhabt werden.

-R **Browser Referenz Informationen in den OBJs** schließen Browser Informationen ein, wenn der Compiler OBJ.Dateien erzeugt. Dies gibt Ihnen die Möglichkeit, sich Ihr Programm über den in der IDE integrierten Browser anzusehen. Wenn diese Option ausgeschaltet ist, können Sie größere .OBJ-Dateien linken. Diese Option hat keinen Einfluß auf die Ausführungsgeschwindigkeit, aber auf die Compilierdauer.

Compiler/Vorcompilierte Header

-H **Erzeugen und verwenden** erzeugt und verwendet vorcompilierte Header. Dazu wird der Standarddateiname BCDEF.CSM (16-Bit) oder BC32DEF.CSM (32-Bit) für den Kommandozeilencompiler und <Dateina-

me>.CSM für Projekte in der IDE verwendet. Vorcompilierte Header können die Compiliergeschwindigkeit drastisch erhöhen, obwohl sie beträchtlichen Plattenspeicher benötigen.

-Hu **Verwenden, aber nicht erzeugen** verwendet vorcompilierte Header, legt diese aber nicht an.

-H- **Nicht erzeugen oder verwenden** legt keine vorcompilierten Header an und verwendet keine.

-H=Dateiname **Vorcompilierte Header-Dateinamen** legen vorcompilierten Header an und verwenden diese. Diese Option gibt der Datei für die vorcompilierten Header den Namen (anders als BCDEF.CSM oder BC32DEF.CSM).

-H»xxx« **Stop vorcompilieren nach Header-Datei** beendet das Compilieren der vorcompilierten Header, wenn die Datei als xxx spezifiziert ist.

16-Bit Compiler/Prozessor

-2 **80286** erzeugt 16-Bit 80286 Protected-Mode-kompatible Anweisungen.

-3 **80386** erzeugt 16-Bit 80386 Protected-Mode-kompatible Anweisungen.

-4 **i486** erzeugt 16-Bit 80486 Protected-Mode-kompatible Anweisungen.

-an **Datenausrichtung Byte/Word** richtet folgendermaßen aus n: 1 = **Byte**, 2 = **Word** (2-bytes).

16-Bit Compiler/Aufrufkonvention

-pc -p- **C** erzeugt alle Unterroutinenaufrufe und alle Funktionen, indem es die C-Aufruf-Konvention verwendet. Dies ist gleichbedeutend mit der Deklaration aller Unterroutinen und Funktionen mit dem **__cdel** Schlüsselwort. Die daraus resultierenden Aufrufe sind kleiner und schneller. Funktionen müssen die korrekte Anzahl und den korrekten Typ von Argumenten haben. Dies ist ungewöhnlich im C-Gebrauch, der einem sonst eine beliebige Anzahl von Funktionsargumenten erlaubt. Sie können als Schlüsselwörter **__cdel**, **__pascal** oder **__fastcall** verwenden, um speziell eine Funktion oder Unterroutine zu deklarieren, indem Sie eine andere Aufrufkonvention verwenden.

-p **Pascal** erzeugt alle Unterroutinenaufrufe und Funktionen, indem es die Pascalaufrufkonvention verwendet. Dies ist gleichbedeutend mit der Deklaration aller Unterroutinen und Funktionen mit dem Schlüsselwort **__pascal**. Die daraus resultierenden Aufrufe sind kleiner und schneller.Funktionen müssen die korrekte Anzahl und den korrekten Typ von Argumenten haben.Sie können als Schlüsselwörter **__cdel**, **__stdcal** oder

__fastcall verwenden, um speziell eine Funktion oder Unterroutine zu deklarieren, indem Sie eine andere Aufrufkonvention verwenden.

-pr **Register** erzeugt alle Unterroutinenaufrufe und Funktionen, indem es die Registeraufrufkonvention verwendet. Dies ist gleichbedeutend mit der Deklaration aller Unterroutinen und Funktionen mit dem Schlüsselwort **__fastcall**. Sie können als Schlüsselwörter **__pascal,** __stdcal oder **__cdecl** verwenden, um speziell eine Funktion oder Unterroutine zu deklarieren, indem Sie eine andere Aufrufkonvention verwenden.

16-Bit Compiler/Speichermodell

Mit den Speichermodelloptionen können Sie dem Compiler mitteilen, welches Speichermodell Sie verwenden, wenn Sie eine 16-Bit Anwendung compilieren (32-Bit Anwendungen sind immer flache Modelle). Die verfügbaren Modelle sind *small*, *medium*, *compact* und *large.*

-ms -ms! **Small** compiliert mit dem Speichermodell Small. (Voreinstellung).

Die Kommandozeilenoption **-ms!** compiliert mit dem Speichermodell *Small* und setzt DS! = SS. Um dies in der IDE zu machen, müssen Sie zwei Optionen überprüfen (Small und Never).

-mm -mm! **Medium** compiliert mit dem Speichermodell *Medium.*

Die Kommandozeilenoption **-mm!** compiliert mit dem Speichermodell *Medium* und setzt DS! = SS. Um dies in der IDE zu machen, müssen Sie zwei Optionen überprüfen (*Medium* und *Never*).

Der tatsächliche Effekt von **-mt!, -ms!** und **-mm!** Optionen ist momentan noch sehr gering. Wenn Sie die Stackvariablenadresse nehmen (Parameter oder auto), wird standardmäßig (wenn gilt : DS ==SS) der resultierende Zeiger zu einem near (relativ zu DS) Zeiger gemacht. Dadurch ist es in diesem Modell ohne Probleme möglich, diese Adresse einem Zeiger mit voreingestellter Größe zuzuweisen. Ist DS != SS, wird bei Verwendung der Adresse einer Stack-Variablen ein **_ss**-Zeiger erzeugt, das heißt der Zeiger kann einem FAR-Zeiger oder einem **_ss**-Zeiger zugewiesen werden bzw. übergeben werden. In den betroffenen Speichermodellen hat aber die Zuweisung einer solchen Adresse an einen **near**-Zeiger oder einen Zeiger mit voreingestellter Größe die Warnung »Suspicious pointer conversion« zur Folge. Derartige Warnungen (sie sind standardmäßig eingeschaltet) zeigen meistens einen Fehler an.

-mc **Compact** compiliert mit dem Speichermodell *Compact.*

-ml **Large** compiliert mit dem Speichermodell *Large.*

-mh **Huge** compiliert mit dem Speichermodell *Huge.*

-mt **Tiny** compiliert mit dem Speichermodell *Tiny.*

-mt! **Tiny** compiliert mit dem Speichermodell *Compact*, wobei DS! = SS

Standard für Modell verwendet das Modell um festzustellen, ob das Stacksegment gleich dem Datensegment ist.

Never geht davon aus, daß das Datensegment niemals gleich dem Stacksegment ist, unabhängig vom Speichermodell.

-Fs **Always** geht davon aus, daß DS in allen Speichermodellen gleich SS ist. Verwenden Sie diese Option, wenn Sie Code portieren, der ursprünglich für eine Implementierung geschrieben wurde, die den Stack als Teil des Datensegments festlegt.

-dc **Strings in Codesegmente ablegen** bewegt alle Stringkonstanten vom Datensegment zum Codesegment der erzeugten Objektdatei, wobei die Daten in den Typ **const** (nur 16-Bit) umgewandelt werden.Diese Option spart Platz für Datensegmente. In großen Programmen, speziell in denen, die eine große Anzahl Stringkonstanten enthalten, wird so die Last von den Datensegmenten auf die Codesegmente geschoben.

Ff **Automatische FAR Daten** ändert globale Variablen, die größer oder gleich der Schwellengröße sind, in **FAR** Variablen. Die Voreinstellung für die -Schwellengröße beträgt 32,767. Diese Option ist für Code nützlich, der nicht das Speichermodell *Huge* einsetzt, der jedoch globale Variablen deklariert, die eine Gesamtgröße von 64 KB überschreiten (oder knapp darunterliegen). Bei den drei Speichermodellen *Tiny*, *Small* und *Medium* hat diese Option keinen Effekt. Wenn Sie diese Option in Verbindung mit -Fc verwenden, sind die erzeugten COMDEFs in den Speichermodellen *Compact*, *Large* und *Huge* vom Typ **FAR**.

Vf **FAR virtuelle Tabellen** bewirkt, daß virtuelle Tabellen in dem Codesegment statt in dem Datensegment angelegt werden (solange nicht die Optionen **-zV** und **-zW** verwendet wurden) und definiert Zeiger auf virtuelle Tabellen als 32-Bit-Zeiger (letzteres wird automatisch durchgeführt, wenn Sie das Speichermodell *Huge* einsetzen).

Es gibt zwei wichtige Gründe, die für die Verwendung dieser Option sprechen: Die virtuellen Tabellen werden nicht im Datensegment, das zu voll werden kann, angelegt und Module, die unterschiedliche Datensegmente verwenden, können Objekte (von Klassen mit virtuellen Funktionen) gemeinsam nutzen (zum Beispiel eine dynamische Linkbibliothek und eine .EXE-Datei, die diese Linkbibliothek einsetzt). Dafür müssen alle Module, die Objekte gemeinsam nutzen, gleich compiliert werden - entweder mit oder ohne diese Option. Dasselbe Ergebnis erreichen Sie, wenn Sie die Modifizierer **huge** oder **_export** klassenweise einsetzen.

-h **Fast huge-Zeiger** bietet eine alternative Methode zur Berechnung von huge-Zeiger Ausdrücken. Diese Methode ist sehr viel schneller als die Standardmethode, aber man muß sie mit Bedacht anwenden. Wenn Sie diese Option verwenden, werden huge-Zeiger dann normalisiert, wenn ein Segmentüberlauf im Offsetteil auftritt. Dies verursacht Probleme mit huge-Arrays, wenn irgendein Arrayelement über die Segmentbegrenzung kommt. Diese Option ist standardmäßig ausgeschaltet.

Normalerweise normalisiert Borland C++ huge-Zeiger, immer wenn von ihm addiert oder subtrahiert wird. Dadurch ist garantiert, daß z.B. , wenn Sie einen huge-Array mit **structs** haben, der größer als 64K ist, das Indizieren in ein Array und Aussuchen eines **struct** Feldes immer mit **structs** jeder Größe gearbeitet wird. Borland C++ gewährleistet dies, indem es immer die Ergebnisse von huge-Zeiger-Operationen normalisiert, so daß der Offsetteil nie eine größere Anzahl als 15 enthält und so ein Segmentüberlauf vermieden werden kann. Der Nachteil dieser Option besteht darin, daß die Ausführungszeit erheblich verlängert wird. Diese Option ist voreingestellt, wenn Sie mit Windows compilieren.

Ff = Größe **FAR Datenschwellengröße** verändert den Punkt, an dem Daten in ein **FAR** umgewandelt werden (verwendet von der -Ff Option).

16-Bit Compiler/Datensegmentnamen für Segmente

Verwenden Sie diese Option nur, wenn Sie die Segmentierung des 80x86-Prozessors kennen. Unter normalen Umständen müssen Segmentnamen nicht vergeben werden.

-zRname **Initialisiertes Datensegment** gibt dem Segment der initialisierten Daten den angegebenen Namen. Die Voreinstellung ist _DATA.

-zSname **Initialisierte Datengruppe** gibt der Gruppe der initialisierten Daten den angegebenen Namen. Die Voreinstellung ist DGROUP.

-zTname **Initialisierte Datenklasse** gibt dem Segment der initialisierten Datenklasse den angegebenen Namen. Die Voreinstellung ist DATA.

-zDname **Uninitialisiertes Datensegment** gibt dem Segment der uninitialisierten Daten den angegebenen Namen. Die Voreinstellung ist _BSS.

-zGname **Uninitialisierte Datengruppe** gibt der Gruppe der uninitialisierten Daten den angegebenen Namen. Die Voreinstellung ist DGROUP.

-zBname **Uninitialisierte Datenklasse** gibt dem Segment der uninitialisierten Datenklasse den angegebenen Namen. Die Voreinstellung ist BSS.

16-Bit Compiler/FAR-Datensegmentnamen

zEname **FAR-Daten Segment** gibt dem Segment, in dem __**FAR**-Objekte gespeichert sind, den angegebenen Namen. Standardmäßig ist der Segmentname der Name der Quelldatei gefolgt von _DATA. Ein Name,der mit einem Sternchen (*) beginnt, zeigt an, daß der Standardstring benutzt werden sollte (nur 16-Bit).

-zHname **FAR-Daten Gruppe** speichert **FAR**-Objekte in der Gruppe mit dem angegebenen Namen. Standardmäßig werden FAR-Objekte nicht in einer Gruppe gespeichert. Ein Name,der mit einem Sternchen (*) beginnt, zeigt an, daß der Standardstring benutzt werden sollte (nur 16-Bit).

-Fname **FAR-Daten Klasse** gibt der Klasse für __**FAR**_Objekte den angegebenen Namen. Die Voreinstellung ist FAR_DATA. Ein Name,der mit einem Sternchen (*) beginnt, zeigt an, daß der Standardstring benutzt werden sollte (nur 16-Bit).

-zVname **FAR Virtuelle FAR-Tabellen Segment** gibt dem Segment für FAR virtuelle Tabellen den angegebenen Namen. Per Voreinstellung werden die FAR virtuellen Tabellen im Code-Segment erzeugt (nur 16-Bit).

-zWname **FAR Virtuelle FAR-Tabellen Klasse** gibt dem Klassensegment für FAR virtuelle Tabellen den angegebenen Namen. Per Voreinstellung werden die FAR virtuellen Tabellen im Code-Segment erzeugt (nur 16-Bit).

16-Bit Compiler/Namen für Code-Segmentnamen

-zCname **Code-Segment** gibt dem Code-Segment den angegebenen Namen. Die Voreinstellung ist :TEXT.

-zPname **Code-Gruppe** gibt der Gruppe der Code-Segmente in den Ausgabedateien den angegebenen Namen.

-zAname **Code-Klasse** gibt der Code-Segment-Klasse den angegebenen Namen. Standardmäßig ist das Code-Segment der Klasse CODE zugewiesen.

16-Bit Compiler/Entry/Exit-Code

Entry/Exit-Code-Optionen geben an, um welchen Anwendungstyp es sich handelt, den der Compiler erzeugt.

-tW -W -WC- **Windows: alle Funktionen exportierbar** legt die allgemeinste Art einer ausführbaren Windows-Datei an, die jedoch nicht zwangsläufig die effizienteste ist. (-W- ist die Standardoption). Diese Option erzeugt die not-

wendigen Export-Informationen für jede **FAR**-Funktion, unabhängig davon ob die Funktion diese benötigt oder nicht. Sie nimmt an, daß alle Funktionen verfügbar sind für die Aufrufe vom Windows Kernel oder anderen Modulen.

Wenn diese Option von einer 16-Bit Anwendung verwendet wird, erzeugt sie eine Windows.EXE Funktion prolog/epilog für alle **FAR**-Funktionen und stellt sie dann bereit, daß sie von anderen Modulen aufgerufen werden können. Um die Funktionsadresse aktuell von .EXE nach .DLL zu exportieren, schließt der Code einen Aufruf an *MakeProcInstance()* ein, der den resultierenden Zeiger zu der.DLL gibt und die Adresse abfragt. Um die Funktionsadresse aktuell von der .DLL zu exportieren, müssen die Funtktionsnamen in die .DEF-Datei eingebunden sein.

-tWE -WE **Windows: explizite Funktionen exportieren** legt ein Windows Objekt-Modul an, in dem nur die Funktionen exportiert werden, die mit dem Schlüsselwort **_export** markiert sind. Verwenden Sie diese Option, wenn Sie Funktionen haben, die nicht über den Windows Kernel aufgerufen werden. Windows explizite Funktionen Export arbeitet genau wie Windows alle Funktionen exportierbar, außer daß nur die mit dem Schlüsselwort _export markierten Funktionen (und Methoden von Klassen, die mit _export markiert sind) den extra prolog/epilog erhalten.

Diese Option ist weitaus effizienter für 16-Bit Anwendungen als die Windows alle Funktionen exportierbar, da nur die Funktionen, die von außerhalb des Moduls aufgerufen werden, den Overhead des prologs bekommen. Diese Option muß im Vorhinein entscheiden, welche Funktionen/Klassen exportiert werden müssen. MakeProcInstance() wird weiter verwendet, aber eine .DEF-Datei Manipulation ist nicht mehr nötig.

-tWS -WS **Windows intell. Callbacks, alle Funktionen exportierbar** (nur 16-Bit Anwendungen) legt ein Modul an mit Funktionen, die Smart Callbacks verwenden, und alle Funktionen exportiert. Verwenden Sie diese Option nur, wenn der Compiler vermuten kann, daß DS == SS für alle Funktionen in dem Modul gilt (was für die Mehrheit der Windows-Programme zutrifft und Standard ist für Borland-Tools).

Diese Option erzeugt eine Windows EXE Funktion prolog/epilog für alle **FAR**-Funktionen und stellt diese dann abrufbar für andere Module bereit. *MakeProcInstance()* wird nicht weiter verwendet, eine .DEF-Datei Manipulation ist nicht mehr nötig.

-tWSE -WSE **Windows intell. Callbacks, explizite Funktionen exportieren** erzeugt eine 16-Bit Windows-Anwendung mit Small Callbacks und expliziten, exportierbaren Funktionen. Diese Option ist die gleiche wie Windows Smart Callbacks, außer daß nur die mit dem Schlüsselwort _export markierten Funktionen (und Methoden von Klassen, die mit _export markiert sind) den extra prolog/epilog erhalten. Diese Option ist weitaus effizien-

ter für 16-Bit Anwendungen als die Windows alle Funktionen exportierbar, da nur die Funktionen, die von außerhalb des Moduls aufgerufen werden, die den Overhead des prologs bekommen. Diese Option muß im Vorhinein entscheiden, welche Funktionen/ Klassen exportiert werden müssen.

-tWD -WD **Windows DLL, alle Funktionen exportierbar** legt ein Modul für den Einsatz in einer dynamischen Linkbibliothek an, wobei alle Funktionen für den Export bestimmt sind. Diese Option erzeugt eine Windows DLL Funktion Prolog/Epilog für alle **FAR**-Funktionen und stellt diese dann abrufbar für andere Module bereit. Um die Funktionsadresse aktuell von der .DLL zu exportieren, müssen die Funktionsnamen in der .DEF-Datei eingebunden werden.

-tWDE -WDE **Windows DLL, explizite Funktionen exportierben** erzeugt ein DLL Objekt-Modul, in das alle Funktionen, die mit -export markiert sind, exportiert werden. Die Windows DLL, explizite Funktionen exportierbar arbeitet wie die Windows DLL, alle Funktionen exportierbar, außer daß nur die mit dem Schlüsselwort _export markierten Funktionen (und Methoden von Klassen, die mit _export markiert sind) den extra prolog/epilog erhalten. Dies ist weitaus effizienter als Windows DLL, alle Funktionen exportierbar, da nur die Funktionen, die von außen aufgerufen wurden, die den Overhead des prologs bekommen. Diese Option muß im Vorhinein entscheiden, welche Funktionen/ Klassen exportiert werden müssen. Eine .DEF-Datei ist nicht nötig.

32-Bit Compiler/Prozessor

-3 **80386** erzeugt Anweisungen, die zu 32-Bit 80386 Protected-Mode kompatibel sind.

-4 **i486** erzeugt Anweisungen, die zu 32-Bit 80486 Protected-Mode kompatibel sind.

-5 **Pentium** erzeugt Anweisungen, die zu Pentium Protected-Mode kompatibel sind.

-a*n* **Datenausrichtung Byte/Word/Double word** richtet die Befehlszeile nach *n* aus, wobei *n* nach den IDE-Optionen wie folgt definiert ist: 1 = **Byte**, 2 = **Word** (2 Bytes), 4 = **Double word** (4 Bytes). Ausrichtung auf Word erhöht die Geschwindigkeit, mit der 80x86-Prozessoren Daten laden und speichern.

32-Bit Compiler/Aufrufkonvention

-pc -p- **C** erzeugt alle Unterroutinenaufrufe und alle Funktionen unter Verwendung der C-Aufrufkonvention und hat die gleiche Bedeutung, wie die Deklaration aller Unterroutinen und Funktionen mit dem Schlüsselwort **__cdedl**. Die resultierenden Funktionsaufrufe sind kürzer und schneller. Funktionen müssen die korrekte Anzahl und den korrekten Typ von Argumenten übergeben; dies entspricht nicht dem normalen Gebrauch von C, der eine unterschiedliche Anzahl von Funktionsargumenten erlaubt. Um die Verwendung einer anderen Aufrufkonvention für eine Funktion oder Unterroutine speziell zu deklarieren, kann man die Schlüsselwörter **__cdecl**, **__pascal**, **__stdcall** oder **__fastcall** benutzen.

-p **Pascal** erzeugt alle Unterroutinenaufrufe und alle Funktionen unter Verwendung der Pascal-Aufrufkonvention und hat die gleiche Bedeutung, wie die Deklaration aller Unterroutinen und Funktionen mit dem Schlüsselwort **__pascal**. Die resultierenden Funktionsaufrufe sind kürzer und schneller. Funktionen müssen die korrekte Anzahl und den korrekten Typ von Argumenten übegeben. Um die Verwendung einer anderen Aufrufkonvention für eine Funktion oder Unterroutine speziell zu deklarieren, kann man die Schlüsselwörter **__cdecl**, **__stdcall** oder **__fastcall** benutzen.

-pr **Register** erzeugt alle Unterroutinenaufrufe und alle Funktionen unter Verwendung der Register-Aufrufkonvention und hat die gleiche Bedeutung, wie die Deklaration aller Unterroutinen und Funktionen mit dem Schlüsselwort **__fastcall**. Um die Verwendung einer anderen Aufrufkonvention für eine Funktion oder Unterroutine speziell zu deklarieren, kann man die Schlüsselwörter **__stdcall**, **__pascal**, **__fastcall** oder **__cdecl** benutzen.

-ps **Standardaufruf** verwendet stdcall-Aufrufkonventionen. Diese Option weist den Compiler an, Pascalsortierung für das Pushen von Parametern auf den stack vorzunehmen. Parameter werden von links nach rechts gepusht.

C++ Optionen/Elementzeiger

-Vmp **Berücksichtigung der Genauigkeit von Elementzeigern** verwendet die deklarierte Genauigkeit für Elementzeiger-Typen. Diese Option sollte verwendet werden, wenn ein Zeiger auf eine abgeleitete Klasse explizit als Zeiger-auf-Element einer einfacheren Basisklasse gecastet ist (wenn der Zeiger gegenwärtig auf ein abgeleitetes Klassenelement zeigt).

-Vmv **Unterstützung aller Fälle** läßt Elementzeiger auf alle möglichen Elemente zeigen. Elementzeiger verwenden die allgemeinste (aber nicht immer die effizienteste) Darstellung.

-Vmm Unterstützung von Mehrfachvererbung läßt Elementzeiger auf Elemente von Klassen mit Mehrfachvererbung zeigen, außer wenn es sich um Elemente von virtuellen Basisklassen handelt.

-Vms Unterstützung von Einfachvererbung läßt Elementzeiger nur auf Elemente von Klassen mit Einfachvererbung zeigen.

-Vmd **Kleinstmögliche Darstellung der Klasse** verwendet die kleinstmögliche Darstellung, die es Elementzeigern ermöglicht, auf alle Elemente ihrer Klasse zu zeigen. Wenn die Klasse zu dem Zeitpunkt, zu dem der Elementzeiger-Typ deklariert wird, nicht vollständig definiert ist, wählt der Compiler den allgemeingültigsten Typ für den Zeiger aus. Er gibt dabei eine entsprechende Warnung aus.

C++ Optionen/C++ Kompatibilität

-K2 **Nur zwei Zeichen-Typen sind zugelassen** behandelt **char** als vorzeichenbehaftet. Kompatibel mit Borland C++ 3.1 und früheren Versionen (nur 16-Bit).

-Vb- Immer Near läßt den Compiler einen verborgenen Zeiger als near speichern. Wenn eine Klasse virtuell von einer Basisklasse erbt, speichert der Compiler einen verborgenen Zeiger im Klassenobjekt, um den Zugriff auf das Unterobjekt der virtuellen Basisklasse zu ermöglichen. Diese Option erzeugt kürzeren und effizienteren Code.

-Vb Gleiche Größe wie der 'this'-Zeiger vergleicht den verborgenen Zeiger mit der Größe des 'this'-Zeigers der eigenen Klasse.

-Va Klassenwerte als Referenz übergeben Wenn ein Argument vom Typ 'Klasse mit Konstruktoren' über den Wert an eine Funktion übergeben wird, weist diese Option den Compiler an, beim Aufrufer eine temporäre Variable zu erzeugen, diese Variable mit dem Argumentwert zu initialisieren und eine Referenz auf diese temporäre Variable an die Funktion zu übergeben.

-Vc **Konstruktorersetzungen abschalten.** Diese Option weist den Compiler an, per Voreinstellung keine verborgenen Elemente und keinen verborgenen Code zur abgeleiteten Klasse hinzuzufügen. Dies stellt die Kompatibilität zu früheren Compilerversionen sicher.

-Vp **echte 'Pascal'-Elementfunktionen** weist den Compiler an, den Parameter 'this' als ersten Parameter auf dem Stack an 'Pascal'-Elementfunktionen zu übergeben. Per Voreinstellung bringt der Compiler in der Version

3.0 'this' immer als letzten Parameter auf den Stack, unabhängig von der Aufrufkonvention.

-Vv **Tiefe virtuelle Basisklassen.** Wenn eine abgeleitete Klasse eine virtuelle Funktion überschreibt, die sie von einer virtuellen Basisklasse geerbt hat, und ein Konstruktor oder Destruktor der abgeleiteten Klasse diese virtuelle Funktion unter Verwendung eines Zeigers auf die virtuelle Basisklasse aufruft, dann kann es vorkommen, daß der Compiler verborgene Elemente zu der abgeleiteten Klasse hinzufügt, und daß er zusätzlichen Programmcode zu seinem Kostruktor und seinem Destruktor hinzufügt. Diese Option weist den Compiler an, **keine** verborgenen Elemente und **keinen** zusätzlichen Programmcode hinzuzufügen, so daß das Erscheinungsbild der Klasseninstanz dem in früheren Versionen entspricht.

-Vt Zeiger auf virtuelle Tabelle hinter den Datenelementen weist den Compiler an, den virtuellen Tabellenzeiger nach nicht-statischen Datenelementen der betreffenden Klasse zu plazieren, um die Kompatibilität zu gewährleisten, wenn Klasseninstanzen auch von nicht-C++ Code verwendet werden und Klassen gemeinsam genutzt werden, deren Programmcode mit früheren Borland C++ Versionen compiliert wurde.

C++-Optionen/Virtuelle Tabellen

Die Option -Vn steuert die virtuellen C++-Tabellen. Es stehen folgende Möglichkeiten zur Verfügung:

-V **Automatisch** legt virtuelle C++-Tabellen zwischen den Modulen einer Applikation an. **inline**-Funktionen werden dabei nicht inline erweitert. Auf diese Weise wird nur eine Instanz einer virtuellen Tabelle oder **inline**-Funktion im Programm erzeugt. Dadurch werden sehr kleine und effiziente .EXE-Dateien erzeugt, jedoch sind hierfür .OBJ- und .ASM-Namenserweiterungen nötig, die aber nur bei TLINK und TASM zur Verfügung stehen.

-Vs **Lokal** legt lokal virtuelle Tabellen an (Inline-Funktionen werden nicht inline erweitert), so daß jedes Modul über eine eigene Kopie jeder virtuellen Tabelle (oder nicht-inline erweiterten **inline**-Funktion) verfügt. Diese Option erzeugt etwas längere .EXE-Dateien als die Option Smart.

-V0 **Extern** erzeugt externe Referenzen auf virtuelle Tabellen. Wenn Sie nicht die Smart- oder Lokal-Optionen verwenden wollen, können Sie mit Extern und Public globale virtuelle Tabellen anlegen und referenzieren.

-V1 **Public** erzeugt öffentliche Definitonen für virtuelle Tabellen. Bei Einsatz der beiden Optionen Extern und Public muß mindestens ein Programmodul mit der Public-Option compiliert werden, damit die Definitionen für virtuelle Tabellen angelegt werden. Alle anderen Module sollten jedoch

mit der Option **-V0** compiliert werden, um auf die Public-Kopie der virtuellen Tabellen Bezug zu nehmen.

C++-Optionen/Templates

-Jg **Automatisch** erzeugt öffentliche Definitionen von allen Template-Instanzen. Wenn dieselbe Template-Instanz von mehreren Modulen erzeugt wird, erstellt der Linker aus diesen eine einzelne Kopie der Instanz. Diese Option setzt voraus, daß dem Compiler für Template-Funktionen der Funktionsrumpf und für Template-Klassen der Rumpf der Elementfunktionen und die Definitionen der statischen Datenelemente zur Verfügung stehen.

-Jgd **Global** erzeugt für alle Template-Instanzen nummerierte öffentliche Definitionen. Doppelte Instanzen werden nicht zusammengefaßt. Ist dieselbe Template-Instanz in mehreren Modulen definiert, gibt der Linker eine Fehlermeldung über die Neudefinition eines öffentlichen Symbols aus.

-Jgx **Extern** erzeugt externe Referenzen auf Template-Instanzen. Sie können diese Option nur dann verwenden, wenn die Instanzen vorher in einem anderen Modul mit der **-Jgd**-Option als öffentlich definiert worden sind, damit die externen Referenzen erstellt werden können.

C++- Optionen/Exceptions/RTTI

-x **Ausnahmebehandlung anfordern** schaltet C++-Ausnahmebehandlung ein. Wenn Sie Ausdrücke zur C++-Ausnahmebehandlung in Ihrem Programmcode verwenden und *nicht* mit dieser Option compilieren (durch Nicht-Prüfung der Option in der IDE oder durch Verwendung der Befehlszeilen-Option **-x-**) führt dies zu einer Fehlermeldung.

-xp **Informationen zur Suche von Ausnahmen anfordern** identifiziert während der Laufzeit auftretende Ausnahmen indem sie die Zeilennummer des Quelltextes angibt, in der die Ausnahme aufgetreten ist. Diese Option läßt das Programm die Datei und die Zeilennummer ausgeben, in der die C++-Ausnahme aufgetreten ist.

-xd **Aufräumen des Destruktors anfordern.** Nach dem Auftreten und vor der Ausgabe der Ausnahme werden für alle automatisch deklarierten Objekte die Destruktoren aufgerufen. Beachten Sie, daß die Destruktoren für dynamische Objekte *nicht* aufgerufen werden und dynamische Objekte somit nicht automatisch freigegeben werden.

-RT **Typinformationen zur Laufzeit anfordern** erzeugt Code, der Informationen über die Typen zur Laufzeit beinhaltet.

Optimierungen

Der Borland Compiler enthält Optimierungen zur Erhöhung der Ausführungsgeschwindigkeit einer Anwendung und zur Reduktion ihrer Größe. Der Compiliervorgang dauert nur 50% länger um die Ausführungsgeschwindigkeit zu optimieren und 20% länger um die Größe zu optimieren. Sie können zu jeder Zeit während Ihres Projektzyklus' mit den Optimierungsoptionen compilieren. Bei der Fehlersuche kann es manchmal hilfreich sein, diese Optionen eingeschaltet zu haben, da der integrierte Debugger mit optimiertem Code arbeitet.

-Od **Alle Optimierungen abschalten** schaltet alle Optimierungen aus. Sie können diese Option überschreiben, indem Sie vorbenannte Optionsmengen im Projekt-Manager verwenden.

Optimierungen/Spezielle Optimierungen

-O2 -Ot -Ox -G **Programmgeschwindigkeit** erzeugt den schnellst möglichen Code. Der Compiler überprüft, ob er problemlos Code erzeugen kann, um eine **rep movsw**-Anweisung darzustellen, statt durch Aufruf einer Hilfsfunktion diese zu kopieren. Dieses erzeugt schnellere Struktur-Kopien für Strukturen und Varianten mit einer Länge von über 8 Byte, als der Aufruf der Hilfsfunktion. Die Befehlszeilen-Option **-Ox** dient der Kompatibilität mit dem Microsoft Compiler.

-O1 -Os -G- **Programmgröße** erzeugt den kleinstmöglichen Code, indem der erzeugte Code nach doppelten Sequenzen abgesucht wird. Wenn eine doppelte Sequenz auftritt, ersetzen die Optimierungen eine der Sequenzen durch einen Sprung auf die andere und löscht sie. Dies kommt am häufigsten in Zusammenhang mit der **switch**-Anweisung vor.

Nicht optimieren für allgemeine Unterausdrücke.

-Oc **lokal Optimieren** löscht allgemeine Unterausdrücke in Anweisungsgruppen ohne Sprunganweisungen (Basisblöcke).

-Og **global Optimieren** löscht doppelte Ausdrücke innerhalb des Zielbereichs und speichert die berechneten Werte dieser Ausdrücke einmal (statt sie neu zu berechnen). Obwohl diese Optimierung theoretisch die Größe des Programmcodes reduzieren könnte, bewirkt sie eine Geschwindigkeitsoptimierung und selten eine Größenreduzierung. Verwenden Sie diese Option, wenn Sie Ausdrücke lieber wiederverwenden, als sie für jede Instanz neu zu berechnen.

-Oa **Kein Alias-Namen für Zeiger** voraussetzen hat Einfluß auf die Art, in der die Optimierungen das Löschen allgemeiner Unterausdrücke und

den Ersatz von Variablen durch Ausdrücke durchführt, indem die Optimierungen Informationen über den Ersatz von Variablen durch Ausdrücke über Funktionsaufrufe hinaus bewahrt. Außerdem bewahren die Optimierungen Informationen über allgemeine Unterausdrücke einige Speichervorgänge lang auf. Ohne diese Option sind die Optimierungen gezwungen, Informationen über Kopien und Unterausdrücke zu verwerfen. Alias-Namen für Pointer können Fehler verursachen, die sehr schwer zu finden sind, aus diesem Grund sind Alias-Namen für Pointer nur möglich, wenn die Option **-Oa** benutzt wird.

-Oa steuert die Behandlung von Ausdrücken, die Pointer enthalten, durch die Optimierungen. Wenn mit allgemeinen Unterausdrücken und der eingeschalteten Option **-Oa** compiliert wird, erkennt der compiler `*p * x` in **func1** als einen allgemeinen Unterausdruck.

```
int g, y;
int func1(int *p)
{
   int x=5;
   y = *p * x;
   g = 3;
   return (*p * x);
}
void func2(void)
{
   g = 2;
   func1(&g);
   // Dies ist ein Fehler--Die Zuweisung g = 3
   // macht den Ausdruck *p * x ungültig
}
```

Optimierungen/Größe

-O **Sprungoptimierung** optimiert Sprünge. Wenn diese Option eingeschaltet ist, verkleinert der Compiler die Größe des Programmcodes durch Löschen von redundanten Sprüngen und durch Reorganisierung von Schleifen und *switch*-Anweisungen. Wenn diese Option eingeschaltet ist, können Trace- und Step-Sequenzen leicht zu Verwirrung führen, da es möglich ist, daß mehrere Zeilen des Quelltextcodes auf ein gemeinsames Stück der erzeugten Code-Sequenz abgebildet werden. Wenn diese Option ausgeschaltet ist, bekommen Sie die besten Step-Ergebnisse bei der Fehlersuche.

-Ol **Schleifenoptimierung** nutzt die String-Move-Anweisungen der 80x86 Prozessoren, indem der Code für eine Schleife durch eine String-bewegen-Anweisung ersetzt wird. Dies macht den Code schneller.

```
int v[100];

void t(void)

{

   int i;
   for (i = 0; i < 100; i++)
   v[i] = 0;

}
```

Abhängig von der Komplexität der Operanden kann der kompakte Schleifencode kleiner sein als der entsprechende nicht-kompakte Code.

-Z **Keine redundanten Ladeoperationen** , das Sie immer verwenden sollten, wenn sie mit Optimierungen compilieren, hilft bei der Optimierung sowohl der Geschwindigkeit, als auch der Codegröße, unter Beachtung der Werte, die in den Registern stehen. Werte, die bereits in Registern stehen werden nicht noch einmal geladen.

-Ob **Entfernung überflüssiger Anweisungen** bringt die Variablen zum Vorschein, die nicht mehr gebraucht werden. Da der Optimierer entscheiden muß, wann Variablen nicht weiter genutzt werden, und wann ihr Wert gebraucht wird (Lebensdauer-Analyse), müssen Sie die Globale Register-Reservierung (Option **-Oe**) benutzen, wenn die Option **-Ob** eingeschaltet ist.

-OW **Windows-Prolog/Epilog** unterdrückt die *bp / dec bp inc bp / dec bp des Prolog- bzw. Epilogcodes einer Windows Far-Funktion. Wenn die Debugger-Option (***-v**) eingeschaltet ist, ist diese Funktion ausgeschaltet, weil einige Tools zur Fehlersuche (z.B. WinSpector oder Turbo Debugger für Windows) den inc bp / dec bp benötigen um die Stack-Frame-Informationen anzuzeigen.

-Oe **Globale Registerzuweisung** sollten Sie immer verwenden, wenn Sie Code optimieren möchten. Diese Option erhöht die Ausführungsgeschwindigkeit und verkleinert die Größe Ihrer Anwendung. Wenn die Option -Oe eingeschaltet ist, ist sowohl die globale Registerreservierung, als auch die Lebensdauer-Analyse für Variablen aktiviert.

Optimierungen/Geschwindigkeit

-Oi **Intrinsic-Funktionen als Inline** erzeugen den Code für Speicherfunktionen (z.B. **strcpy** oder **memcmp**) innerhalb des Funktionsbereichs, dadurch entfällt die Notwendigkeit für einen Funktionsaufruf. Der erzeugte

Code ist schneller, aber auch umfangreicher. Im folgenden sind die Funktionen aufgeführt, die bei aktivierter **-Oi**-Option inline erweitert werden:

- fabs
- memchr
- memcmp
- memcpy
- memset
- rotl
- rotr
- stpcpy
- strcat
- strchr
- strcmp
- strcpy
- strlen
- strncat
- strncmp
- strncpy
- strnset
- strrchr
- alloca

Mit #pragma **intrinsic** können Sie die inline-Erweiterung dieser Funktionen für jeden Einzelfall getrennt steuern. Beispielsweise würde #pragma intrinsic strcpy bewirken, daß der Compiler Code für **strcpy** in Ihrer Funktion erzeugt. #pragma intrinsic -strcpy verhindert, daß der Compiler **strcpy** inline erweitert. Wenn Sie diese Pragmas in einer Datei einsetzen, werden die Kommandozeilen- oder die IDE-Optionen vorübergehend außer Kraft gesetzt, die für die Compilierung dieser Datei verwendet werden.

Wenn Sie eine intrinsic-Funktion inline erweitern, müssen Sie vor ihrer Verwendung für diese Funktion einen Prototyp angeben. Dies ist deshalb notwendig, weil der Compiler ein Makro erzeugt, das die inline erweiterte Funktion in eine andere, vom Compiler intern definierte Funktion, umbenennt. Im obigen Beispiel würde der Compiler folgendes Makro erzeugen: #define strcpy __strcpy__.

Der Compiler erkennt Funktionsaufrufe mit zwei vorangestellten und zwei nachgestellten Unterstrichen und versucht, den Prototyp dieser Funktion mit seinem eigenen internen Prototyp in Übereinstimmung zu bringen. Wenn Sie keinen Prototyp angeben oder der angegebene Prototyp nicht mit dem internen Prototyp des Compilers übereinstimmt, wird die Funktion nicht inline erweitert und der Compiler gibt eine Fehlermeldung aus.

-Om **Verschiebung von invariantem Code** aus Schleifen verschieben. Das Verschieben von invariantem Code aus Schleifen erhöht die Ausführungsgeschwindigkeit. Bei der Optimierung werden Informationen über alle Ausdrücke in der Funktion verwendet, die bei der Eliminierung der gemeinsamen Unterausdrücke gesammelt wurden, um diejenigen Ausdrücke herauszufinden, deren Werte in einer Schleife nicht geändert werden dürfen. Um zu verhindern, daß innerhalb einer Schleife Berechnungen mehrfach ausgeführt werden, wird bei der Optimierung der Code aus der Schleife geschoben, so daß nur eine einzige Berechnung nötig ist. Durch die Optimierung wird der berechnete Wert mehrfach in der Schleife verwendet. Sie sollten immer dann invarianten Code aus Schleifen schieben lassen, wenn Sie mit der Ausführungsgeschwindigkeits-Optimierung compilieren und die Option zur Eliminierung gemeinsamer Un-

terausdrücke verwendet haben. Das Verschieben von Code aus Schleifen kann zu einer enormen Erhöhung der Ausführungsgeschwindigkeit führen. Im folgenden Beispiel wird *x*y*z* in jeder Schleifenwiederholung ausgewertet:

```
int v[10];

void f(void)
{
   int i,x,y,z;
   for (i = 0; i < 10; i++)
   v[i] = x * y * z;
}
```

Durch die Optimierung wird der Code für die Schleife folgendermaßen umgeschrieben:

```
int v[10];

void f(void)
{
   int i,x,y,z,t1;
   t1 = x * y * z;
   for (i = 0; i < 10; i++)
   v[i] = t1;
}
```

-Op **Kopien verwenden.** Die Wiederverwendung von Werten für Kopien dient in erster Linie der Erhöhung der Ausführungsgeschwindigkeit. Da der Umfang des Programms niemals erhöht wird, sollten Sie diese Optimierung bei aktivierter *-Og*-Option verwenden. Wie das Verschieben von invariantem Code aus Schleifen beruht auch die Wiederverwendung von Werten für Kopien auf der während der Eliminierung von Unterausdrücken durchgeführten Analyse. Werte für Kopien wiederverwenden bedeutet, daß der Compiler sich bei der Optimierung an die den Ausdrücken zugewiesenen Werte erinnert und diese Werte verwendet, anstatt den Wert der zugewiesenen Ausdrücke zu laden. Werte können für Kopien von Konstanten, Ausdrücken und Variablen wiederverwendet werden.

-Ov **Entfernung von Induktionsvariablen** erzeugt Induktionsvariablen zur Verminderung der CPU-Last und optimiert die Ausführungsgeschwindigkeit für Schleifen. Bei der Optimierung wird ein mathematisches Vorgehen eingesetzt, das Induktion genannt wird, um damit neue Variablen aus Ausdrücken zu erzeugen, die in Scheifen verwendet werden. Diese Variablen heißen Induktionsvariablen. Die Optimierung stellt sicher, daß die mit diesen neuen Variablen ausgeführte Operationen weniger re-

chenintensiv sind (Verminderung der CPU-Last) als diejenigen, die mit den ursprünglichen Variablen ausgeführt werden.

Diese Optimierungen werden normalerweise dann verwendet, wenn Sie in Schleifen Arrays indizieren. In diesem Fall ist eine Multiplikationsoperation erforderlich, mit deren Hilfe die von der Indexnummer angegebene Position im Array berechnet wird. Durch die Optimierung würde z.B. im folgenden Code aus der Operation *v[i]* eine Induktionsvariable erzeugt werden, da die *v[i]*-Operation eine Multiplikation erfordert. Diese Induktionsvariable verhindert auch, daß der Wert von *i* gesichert werden muß.

```
int v[10];
void f(void)

{
   int i,x,y,z;
   for (i = 0; i < 10; i++)
 v[i] = x * y * z;
}
```

Wenn -Ov aktiviert ist, würde durch die Optimierung Code erzeugt, der folgendem Programmtext entspricht:

```
int v[10]
void f(void)

{
   int i,x,y,z, *p;
   for (p = v; p < &v[10]; p++)
   *p = x * y * z;

}
```

Warnungen

Mit einem Stern (*) gekennzeichnete Warnungen sind standardmäßig aktiviert. Diese Optionen werden von der Entwicklungsumgebung (IDE) überprüft.

-w **Alle** zeigt alle Warnungen an.

-wxxx -w-xxx **Ausgewählte** aktiviert bzw. unterdrückt (-w-xxx) die Warnung xxx. Die möglichen Optionen für -wxxx werden im folgenden aufgelistet. Sie lassen sich in vier Kategorien einteilen: *ANSI Violations*, *Frequent Errors*, *Portability Warnings* und C++ *Warnings*. Die Kontrolle dieser Optionen erfolgt im Quelltext mit dem Pragma **warn**. Näheres dazu finden Sie in den Erläuterungen der Präprozessor-Direktiven im *Programmierhandbuch*.

-w- **Keine** zeigt keine Warnungen an. Fehler werden weiterhin angezeigt.

-g*n* **Beenden nach n Warnungen** läßt den Compiler nach n Warnungen anhalten, die in Ihrem Projekt auftreten.

-j*n* **Beenden nach n Fehlern** läßt den Compiler nach n Fehlern anhalten, die in Ihrem Projekt auftreten.

Warnungen/Portabilität

-wrpt* Nonportable pointer conversion.

-wcpt* Nonportable pointer comparision.

-wrng* Constant out of range in comparision.

-wcln Constant is long.

-wsig Conversion may lose significant digits.

-wucp Mixing pointers to different 'char' types.

Warnungen/ANSI Violations

-wvoi* Void functions may not return a value.

-wret* Both return and return with a value used.

-wsus* Suspicious pointer conversion.

-wstu* Undefined structure *Struktur*.

-wdup* Redefinition of *Makro* is not identical.

-wbig* Hexadecimal value contains more than 3 digits.

-wbbf Bit fields must be **signed** or **unsigned int**.

-wext* *Bezeichner* is declared as both external and static.

-wdpu* Declare *Typ* prior to use in prototype.

-wzdi* Division by zero.

-wbei* Initializing *Bezeichner* mit *Bezeichner*.

-wpin Initialization is only partially bracketed.

-wnak Non-ANSI keyword used: *Wort*.

Warnungen/Obsolete C++

-wobi* Base initialization without a class name is now obsolete.

-wofp*	Style of function definition is now obsolete.
-wpre*	Overloaded prefix operator used as a postfix operator.

Warnungen/Potentielle C++-Fehler

-wnci*	Constant member *Bezeichner* is not initialized.
-weas*	Zuweisung von *Typ* an Aufzählungstyp.
-whid*	*Funktion1* hides virtual function *Funktion2*.
-wncf*	Non-const function *Funktion* called for const object.
-wibc*	Base class *Basisklasse1* is also a base class of *Basisklasse2*.
-wdsz*	Array size for 'delete' ignored.
-wnst*	Use qualified name to access nested type *Typ*.
-whch*	Handler for *xxx* is hidden by previous handler for *yyy*.
-wmpc*	Conversion to *Typ* will fail for members of virtual base *Basisklasse*.
-wmpd*	Maximum precision used for member pointer type *Typ*.
-wntd*	Use '> >' for nested templates instead of '>>'.
-wncf*	Non-volatile function *Funktion* called for volatile object.

Warnungen/Ineffiziente C++-Codierung

-winl*	Funktions containing *Bezeichner* are not expanded inline.
-wlin*	Temporary used to initialize *Bezeichner*.
-wlvc*	Temporary used for parameter in call to *Bezeichner*.

Warnungen/Potentielle Fehler

-wpia*	Possibly incorrect assignment.
-wdef	Possible use of *Bezeichner* before definition.
-wnod	No declaration for function *Funktion*.
-wpro*	Call to function with no prototype.
-wrvl*	Function should return a value.
-wamb	Ambiguous operators need parentheses.
-wccc*	Condition is always true/false.

Warnungen/Ineffiziente Codierung

-waus*	*Bezeichner* is assigned a value that is never used.
-wpar*	Parameter *Bezeichner* is never used.
-wuse	*Bezeichner* is declared but never used.
-wstv	Structure passed by Value.
-wrch*	Unreachable code.
-weff*	Code has no effect.

Warnungen/Allgemein

-wasm	Unknown assembler instruction.
-will*	Ill-formed pragma.
-wias*	Array variable *Variable* is near.
-wamp	Superfluous & with function.
-wobs*	*Bezeichner* is obsolete.
-wpch*	Cannot create precompiled header: *Header*.

Make

-w! **Break make on warnings** liefert einen von Null verschiedenen Wert vom Kommandozeilen-Compiler. Diese Meldung tritt auf wenn Warnungen auftraten und keine .OBJ-Datei erzeugt wurde.

Kommandozeilenoptionen

Die im folgenden aufgelisteten Optionen können nur in Verbindung mit den Kommandozeilen-Compilern (BCC.EXE und BCC32.EXE) verwendet werden. Es gibt in der IDE keine Optionen, die direkt äquivalent dazu sind. Weil die meisten Fälle auch in der IDE behandelt werden können, enthält jede Option Anweisungen für die IDE.

-B Compiliert und ruft den TASM zur Bearbeitung von asm-Anweisungen auf. Wenn Sie kein TASM haben, erzeugt diese Option eine Fehlermeldung. Bei früheren Versionen von TASM können Probleme mit dem 32-Bit-Code auftreten.

-c Compiliert und assembliert die bezeichneten .C, .CPP und .ASM Dateien, führt aber kein Link-Kommando aus. Wählen Sie *Projekt/Compiliere* in der IDE.

-e*Dateiname* Benennt die .EXE-Datei mit *Dateiname*.EXE. Dateiname muß direkt auf -e folgen, ohne eingefügte Leerzeichen. Ohne diese Option schreibt der Linker in die .EXE-Datei mit dem Dateinamen der ersten Quelltextdatei oder der ersten Objektdatei aus der Dateinamensliste.

-E*Name* Verwendet den Assembler *Name*. Per Voreinstellung wird mit TASM assembliert. In die IDE können Sie für das Assemblerprogramm, das Sie verwenden möchten, ein Tool einfügen. Siehe Kapitel 2 für Informationen über das Einfügen von Tools in die IDE.

--f87 Benutzt Hardware-Befehle des 80287-Prozessors (Nur 16-Bit).

-f287 Benutzt Hardware-Befehle des 80287-Prozessors (Nur 16-Bit). Wenn Sie ein DOS-Ziel erzeugen wollen, prüfen Sie die Option *Schnelle Gleitkomma-Arithmetik* im TargetExpert.

-Fm Diese Option schaltet alle anderen **-F** Optionen (**-Fc**, **-Ff** und **-Fs**) ein. Sie können diese Option als nützliches Kürzel benutzen, wenn sie Code von anderen Compilern portieren. Um dies in der IDE auszuführen markieren Sie die IDE-Optionen auf **-Fc**, **-Ff** und **-Fs**.

-Fs Stellt sicher, daß in allen Speichermodellen SS=DS ist. Sie können diese Option benutzen, wenn Sie Code portieren, der für eine Implementation geschrieben wurde, die den Stack zu einem Bestandteil des Datensegmentes macht. Wenn Sie diese Option angeben, linkt der Compiler ein alternatives Startmodul (C0Fx.OBJ), das den Stack im Datensegment plaziert. Prüfen Sie *alternatives Startup* im Dialogfeld von TargetExpert. Diese Option arbeitet nur unter DOS.

-Hc Anweisung an den Compiler, vorcompilierte Header anzulegen und wiederzuverwenden. Diese Option muß in Zusammenhang mit **-H** oder **-Hxxx** verwendet werden. Sie ist nützlich, wenn mehr als eine vorcompilierte Header-Datei compiliert werden soll.

-l*x* Gibt die Option x an den Linker (TLINK für BCC und TLINK32 für BCC32) weiter. Nach -l, dem Kleinbuchstaben l, können mehrere Optionen angegeben werden. In der IDE können Sie die Linkeroptionen mittels *Optionen/Projekt/Linker* einstellen. @MARGIN = -l-*x*

Option x wird für den Linker ausgeschaltet. Nach -l-, dem von einem Bindestrich gefolgten Kleinbuchstaben l, können mehrere Optionen angegeben werden. Sie können Linkeroptionen in der IDE mittels *Optionen/Projekt/Linker* setzen und freigeben.

-M Weist den Linker an, eine .MAP-Datei zu erzeugen. Per Voreinstellung wird keine .MAP-Datei erzeugt. Markieren Sie Segment, Öffentlich oder

Detailliert im *Linker/MapFile*-Bereich der Dialogbox *Optionen/Projekt* in der IDE.

-o*Dateiname* Compiliert die Quelltextdatei nach *Dateiname*.obj.

-P- Compiliert Dateien mit der Dateinamenserweiterung .CPP mit C++, andere Dateien werden mit C compiliert. In der IDE gibt es verschiedene Tools, um einen Projektknoten zu compilieren. @MARGIN = -P

Unabhängig von der Dateinamenserweiterung werden alle Dateien als C++-Dateien compiliert. In der IDE gibt es verschiedene Tools, um einen Projektknoten zu compilieren.

-P*ext* Unabhängig von der Dateinamenserweiterung werden alle Dateien als C++-Dateien compiliert. Die Voreinstellung der Dateinamenserweiterung wird auf *ext* gesetzt. Diese Option wird angeboten, da manche Programmierer .C oder andere Dateinamenserweiterungen als Voreinstellung für C++-Code verwenden. In der IDE gibt es verschiedene Werkzeuge, um einen Projektknoten zu compilieren.

-P-*ext* Compiliert abhängig von der Dateinamenserweiterung (.CPP für C++-Compilierung, alle anderen Dateien werden als C-Code compiliert) und setzt die Voreinstellung der Dateinamenserweiterung auf *.ext* (nicht .CPP). In der IDE gibt es verschiedene Werkzeuge, um einen Projektknoten zu compilieren.

-S Erzeugung von Assemblercode compiliert die Quelltextdatei und erzeugt eine .ASM-Datei, assembliert aber nicht. Wenn Sie diese Option verwenden, fügt Borland C++ die C- oder C++-Quelltextzeilen als Kommentare in die erzeugte .ASM-Datei ein. In der IDE gibt es verschiedene Werkzeuge, um einen Projektknoten zu compilieren. Wählen Sie Spezielles/C++ to Assembler aus dem Speed Menü des Projektfensters aus.

-T*Zeichenkette* Übergibt *Zeichenkette* als eine Option an TASM (oder an den mit **-E** ausgewählten Assembler).

-T- Schaltet alle bis zu diesem Zeitpunkt eingeschalteten Assembleroptionen aus.

-tD -tDe Erzeugt eine 16-Bit-Datei für DOS mit der Dateiendung .EXE. Wählen Sie diesen Zieltyp in der IDE, wenn Sie für ihr Projekt ein neues Ziel erzeugen wollen.

-tDc Erzeugt eine 16-Bit-Datei für DOS mit der Dateiendung .COM. Sie können die Dateien mit der Dateiendung .COM nicht in der IDE erzeugen.

-tWC -WC Erzeugt eine 32-Bit-Anwendung für den Textbildschirm-Modus. Wählen Sie diesen Zieltyp in der IDE, wenn Sie ein Ziel für Ihr Projekt erzeugen.

-tWCD -WCD Erzeugt eine 32-Bit-DLL für den Textbildschirm-Modus, deren Funktionen alle exportiert werden. Wählen Sie diesen Zieltyp in der IDE, wenn Sie ein Ziel für Ihr Projekt erzeugen.

-tWCDE -WCDE Erzeugt eine 32-Bit-DLL für den Textbildschirm-Modus, mit explizit exportierten Funktionen. Wählen Sie diesen Zieltyp in der IDE, wenn Sie ein Ziel für Ihr Projekt erzeugen.

-tWM -WM Erzeugt eine Multi-Thread Anwendung oder DLL. Verwenden Sie diese Option mit **-Wm** und **-WCD**. Wählen Sie diesen Zieltyp in der IDE, wenn Sie ein Ziel für Ihr Projekt erzeugen.

-U*name* Macht alle bis zu diesem Zeitpunkt definierten Definitionen des Bezeichners *name* rückgängig.

-Vo Diese Option ist ein Hauptschalter, der alle Optionen für Rückwärtskompatibilität, die in diesem Abschnitt aufgeführt sind, aktiviert. Sie kann als ein nützliches Kürzel eingesetzt werden, wenn mit Bibliotheken gelinkt wird, die mit einer früheren Version von Borland C++ erstellt wurden.

-Y Ermöglicht das generieren von Overlay-Code. Wählen Sie in der IDE DOS Overlay als Zieltyp, wenn Sie einen Zieltyp für Ihr Projekt erzeugen wollen.

-Y0 Overlay für die compilierten Dateien. Prüfen Sie in der IDE das Oberlay dieses Moduls. Im Dialogfeld Knoten-Eigenschaften können Sie prüfen, ob das DOS Overlay-Ziel (-Y) für einen Knoten eingestellt ist.

-zX* Verwendet den Standardnamen für X (Voreinstellung). Z.B. weist **-zA*** der Codesegmentklasse den Namen CODE zu.

K A P I T E L **4**

Anwendungen anlegen mit AppExpert

AppExpert erzeugt ObjectWindows 2.0-Anwendungen, die den Windows-Anwendungen, die Sie erstellen wollen, die Quelltextgrundlagen zur Verfügung stellt. Beispielsweise kann AppExpert den Quelltext für Ihre Haupt- und untergeordneten Fenster erzeugen, für Mauspaletten, Statuszeilen, Menüsysteme und Online-Hilfssysteme. Sie können mit AppExpert auch andere Optionen Ihren Windows-Programmen hinzufügen, wie Bildlaufleisten, die Schaltflächen *Symbol* und *Vollbild* auf einer Titelleiste und *Drucken* und *Druckansicht*.
Nachdem AppExpert eine Anwendung erzeugt hat, können Sie mit Hilfe von ClassExpert und Resource Workshop die Programmfunktionalität vervollständigen. Um den größtmöglichen Nutzen aus AppExpert zu ziehen, sollten Sie mit ClassExpert (in Kapitel 5 beschrieben), Resource Workshop (erläutert in Teil III), ObjectWindows (beschrieben im *ObjectWindows Programmierhandbuch*) und dem Projekt-Manager (siehe hierzu Kapitel 2) vertraut sein.

Mit AppExpert können Sie Programme erstellen, die den folgenden Programmodellen entsprechen:

- Dem Modell für Einzel-Dokumentschnittstellen (SDI), das Ihnen nur das Öffnen eines einzigen Objekts zu einem Zeitpunkt erlaubt.
- Dem Modell für Multi-Dokumentschnittstellen (MDI), das Ihnen ermöglicht, gleichzeitig mehrere Objekte und Ansichten zu öffnen.
- Dem Modell für Dialog Client, das ein Dialogfenster als Haupt-Benutzeroberfläche benutzt.

Außerdem können Sie Ihre Anwendung nach dem Dok/View-Modell von ObjectWindows gestalten (siehe auch das *ObjectWindows Programmierhandbuch* für weitere Information über Dokument/Ansicht).

Dieses Kapitel behandelt die folgenden Abschnitte, um Sie mit AppExpert bekannt zu machen:

- »Grundlagen zu AppExpert« beschreibt die allgemeinen Methoden zur Erzeugung einer Anwendung mit AppExpert.
- »Erstellen einer kleinen Anwendung« zeigt Ihnen, wie eine SDI-Anwendung erstellt wird.
- »Überblick über die Optionen von AppExpert« faßt die Optionen von AppExpert zusammen

Grundlagen zu AppExpert

Die folgenden Schritte zeichnen die Hauptaufgaben auf, die die Erzeugung einer Anwendung betreffen:

1. Benutzen Sie AppExpert, um die Benutzeroberfläche und allgemeine Merkmale Ihrer Anwendung zu definieren.
2. Es erzeugt den Quelltext Ihrer Anwendung.
3. Verwenden Sie ClassExpert, um Ihrer Anwendung Funktionalitäten hinzuzufügen. Mit ClassExpert fügen Sie den Klassen, Ereignisbehandlungsroutinen (Event-Handler), virtuellen Funktionen usw. Codes hinzu.
4. Mit Resource Workshop können Sie Windows-Ressourcen wie Dialoge, Dialogelemente und Menüs hinzufügen und bearbeiten.
5. Benutzen Sie den Projekt-Manager, um Ihr Programm aufzubauen.

Wenn Sie eine Anwendung mit AppExpert erstellen, erzeugt es die folgenden Dateien:

- eine Projektdatei (.IDE)
- eine Quelldatei (.CPP)
- eine Ressourcen-Header-Datei (.RH)
- eine Moduldefinitionsdatei (.DEF)
- eine Header-Datei (.H)
- eine Ressourcenskriptdatei (.RC)
- eine AppExpert-Datenbankdatei (.APX)

Entsprechend den von Ihnen gewählten Optionen kann AppExpert auch folgende Dateien erzeugen:

- Hilfe-Quelldateien (.RTF)
- eine Hilfe-Projektdatei (.HPJ)
- Symbol- und Bitmap-Dateien (.ICO und .BMP)

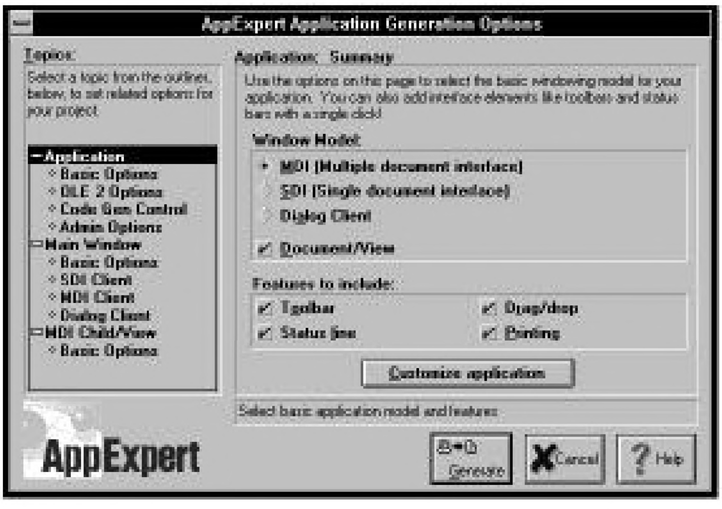
Erstellen einer kleinen Anwendung

Folgendes Beispiel zeigt, wie Sie AppExpert verwenden, um eine SDI-Anwendung zu erstellen, die den ObjectWindows-Rahmen benutzt.

1. Starten Sie die IDE, und wählen Sie *Projekt/AppExpert*, um das Dialogfenster *Neues AppExpert Projekt* zu öffnen.
2. Geben Sie `\myapp\draw` in das Textfeld *Dateiname* ein. Dies beschreibt sowohl den Pfad als auch den Namen der Projektdatei ihrer neuen Anwendung. Standardmäßig werden die meisten erzeugten Dateien (auch .EXE) von dem Projektnamen, den Sie angeben, hergeleitet.

 Wenn das von Ihnen festgelegte Verzeichnis nicht existiert, erzeugt AppExpert es für Sie. Sie können auch das Textfeld *Verzeichnisse* benutzen, um ein existierendes Verzeichnis für Ihr Projekt auszuwählen. In beiden Fällen wird das von Ihnen festgelegte Verzeichnis zum Hauptverzeichnis für alle erzeugten Quelldateien. Für die Organisation Ihrer Projektdateien ist es am besten, ein eindeutiges Hauptverzeichnis für jedes Projekt zu erzeugen.
3. Wählen Sie OK. Das AppExpert-Dialogfenster *Optionen für die Anwendungserzeugung* erscheint. Es enthält auf der linken Seite eine Liste von Themen mit den dazugehörigen Optionen auf der rechten Seite. Um die hierarchische Themenliste zu erweitern klicken Sie die »+« und »-« Symbole auf der linken Seite der aufgelisteten Themen an, oder klicken Sie mit der rechten Maustaste in den Themenausschnitt, um ein lokales Menü zu öffnen, das Befehle zur Erweiterung und Verkürzung der Themenliste anzeigt. Für die Online-Hilfe zu einer der Themen oder Optionen wählen Sie *Hilfe*.

Abbildung 4.1: AppExpert-Dialogfenster *Optionen für die Anwendungserzeugung*



4. Wählen Sie das Thema *Anwendung* und setzen Sie folgende Parameter bei der Option *Zusammenfassung*:
 - Setzen Sie *Fenstermodell* auf *SDI* (Einzel-Dokumentschnittstelle).
 - Heben Sie die Markierung für *Dokument/Ansicht* auf.
 - Heben Sie alle Markierungen in *Einzubindende Eigenschaften* (Mausleiste, Statuszeile, Ziehen/Ablegen und Drucken) auf.

 Diese Optionsparameter beschreiben eine sehr einfache Anwendung. Weitere Informationen über die verfügbaren Anwendungsparameter entnehmen Sie bitte »AppExpert-Optionsreferenz«.
5. Wählen Sie das Thema *Anwendung/Grundoptionen*, dann geben Sie den ersten Buchstaben des Zielnamens als Majuskel (in der Form »Draw«) ein.
6. Wählen Sie das Thema *Anwendung/Optionsverwaltung*, dann ändern Sie den Parameter *Beschreibung* in »Beispielprogramm für ObjectWindows«.
7. Füllen Sie die Textfelder *Autor* und *Firma* aus.
8. Wählen Sie das Thema *Hauptfenster* und setzen Sie *Fenstertitel* auf »Beispielprogramm für ObjectWindows«.
9. Wählen Sie das Thema *Hauptfenster/SDI-Client*, dann setzen Sie *Client/Ansichtsklasse* auf `TWindow` (wählen Sie diese Auswahl aus der Dropdown-Liste).
10. Wählen Sie *Erstellen*, damit AppExpert die Anwendung erzeugt.

 Ein Dialogfenster erscheint und bittet um Bestätigung Ihres Befehls. Wählen Sie *Ja*, wenn AppExpert die Dateien für Ihre Anwendung erzeugen soll. (Wählen Sie *Nein*, wenn Sie noch Optionsparameter ändern müssen.)

 Ein Meldungsfenster zeigt an, wenn AppExpert die Quellmodule, Header-Dateien, Ressource-Dateien und die Projektdatei für Ihre Anwendungen erstellt. AppExpert fügt die Dateien in das Verzeichnis ein, das Sie in der Einstellung *Basisverzeichnis* von *Anwendung / Grundoptionen* festgelegt haben.
11. Nachdem AppExpert die Anwendung erzeugt hat, erscheint das Fenster *Projekt* und listet die meisten der Dateien auf, die AppExpert erzeugt hat. Von hier aus können Sie die Anwendung aufbauen oder mit ClassExpert die Funktionalität der Anwendung hinzufügen. Wählen Sie *Projekt / Neu kompilieren*, um Ihre Anwendung anzulegen.

Anmerkung

Mit AppExpert legen Sie erst Ihre Anwendungsoptionen fest, dann erzeugen Sie den Code. Nach Erzeugung des Codes können Sie die Dateien mit

ClassExpert verändern, Sie können aber nicht zu AppExpert zurückkehren, um Optionen zu verändern. Erzeugen Sie beispielsweise eine Anwendung ohne Statuszeile, können Sie nicht AppExpert benutzen, um später diese Funktionalität hinzuzufügen. Sie müssen es manuell hinzufügen oder die Anwendung von Anfang an neu erzeugen.

Übersicht über den erzeugten Code

AppExpert erzeugt folgende Dateien, wenn Sie das DRAW.IDE Projekt erstellen:

TABELLE 4.1: Anwendungsdateien des Beispiels

Dateiname	Beschreibung
DRAWAPP.CPP	Quelldatei für das Anwendungs- und das Hauptfensterobjekt
DRAWAPP.H	Header-Datei für das Anwendungs- und das Hauptfensterobjekt
DRAWWNDW.CPP	Quelldatei für das Client-Fensterobjekt
DRAWWNDW.H	Header-Datei für das Client-Fensterobjekt
DRWABTDL.CPP	Quelldatei für das Dialogfeldobjekt *Info*
DRWABTDL.H	Header-Datei für das Dialogfeldobjekt *Info*
DRAWAP.RC	Ressourcenskript-Datei für das Projekt
DRAWAP.RH	Ressourcen-Header-Datei für das Projekt
DRAWAP.DEF	Moduldefinitionsdatei für das Projekt
DRAW.APX	Datenbankdateien zur Speicherung der ClassExpert-Projektinformationen
APPLSDI.ICO	Vordefiniertes Symbol für das Projekt
DRAW.IDE	IDE-Projektdatei

Das Anwendungsobjekt

Nachdem Sie DRAW.IDE erzeugt haben, öffnet sich der Projekt-Manager und zeigt den Projektbaum an, der die von AppExpert erzeugten Projektdateien beinhaltet. Abbildung 4.2 stellt DRAW.IDE im *Projekt-Manager* dar (im *Projekt-Manager* sind sowohl die Laufzeitknoten als auch die Projektknoten sichtbar).

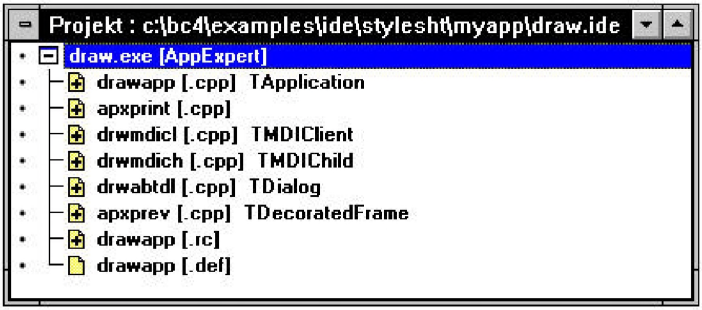


Abbildung 4.2: DRAW.IDE im Projekt-Manager geladen

Vom Projekt-Manager aus öffnen Sie DRAWAPP.CPP durch zweimaliges Anklicken des Listings der Quelldateien. Der übrige Teil dieses Abschnitts beschreibt den Code in dieser Datei.

```
/* Project Draw
  TheSourceWorks, Inc.
  Copyright ã 1994. All Rights Reserved.
  SUBSYSTEM:  draw.exe Application
  FILE:
  AUTHOR:
  :
```

Die ersten Zeilen dieser Datei beschreiben die Anwendung. AppExpert erhält einige dieser Informationen aus Ihren Einträgen im Thema *Anwendung/Optionsverwaltung*. Weiter unten im Listing werden Sie eine Anzahl von Kommentaren mit einer ähnlichen Struktur wie die folgenden Textzeilen sehen:

```
//{{ ...}}
//{{..._BEGIN}}
:
//{{..._END}}
```

Diese zwei Arten von Kommentaren sind ClassExpert-Kommentardatensätze, die von Rescan benutzt werden. Diese Kommentare dürfen nicht geändert werden, wenn Sie vorhaben, ClassExpert bei der Änderung Ihres Projektquelltextes zu verwenden. Sie können zwar nicht Textzeilen ändern, die mit `//{{` beginnen, aber Sie können Textzeilen zwischen die Kommentardatensätze `_BEGIN` und `_END` einfügen, *solange Sie das Format des erzeugten Codes, das sich zwischen diesen Kommentardatensätzen befindet, vervielfältigen*.

In den Header-Dateien, die am Anfang von DRAWAPP.CPP aufgelistet sind, sehen Sie eine Textzeile, die die Datei drawapp.h einschließt. In dieser Header-Datei definiert AppExpert eine Klasse für das Hauptfenster *SDIDecFrame*. In drawapp.h befindet sich auch die Klassendeklaration für Ihr Anwendungsobjekt:

```
⋮
//{{TApplication = DrawApp}}
class DrawApp : public TApplication {
private:

⋮
```

Dieser Code zeigt, daß AppExpert *DrawApp* von *TApplication* ableitet. In *DrawApp* definiert AppExpert einen Konstruktor und Destruktor, überschreibt die Funktion *InitMainWindow* und definiert einige den Kommentar betreffende Kommandofunktionen.

AppExpert definiert die deklarierten Elementfunktionen aus drawapp.h in DRAWAPP.CPP. AppExpert definiert auch eine Antworttabelle, die die Elemente des Hauptmenüs in DRAWAPP.CPP bearbeitet:

```
⋮
//{{DrawApp Implementation}}
//
// Antworttabelle für alle Meldungen/Befehle erzeugen, die von der
// Anwendung behandelt werden.
//
DEFINE_RESPONSE_TABLE1_ (DrawApp, TApplication)
//{{DrawAppRSP_TBL_BEGIN}}
   EV_COMMAND(CM_FILENEW, CmFileNew)
   EV_COMMAND(CM_FILEOPEN, CmFileOpen)
   EV_COMMAND(CM_FILECLOSE, CmFileClose)
   EV_COMMAND(CM_HELPABOUT, CmHelpAbout)
//{{DrawAppRSP_TBL_END}}
END_RESPONSE_TABLE;

⋮
```

Es folgt die Definition des Konstruktors von *DrawApp*. Dieser Konstruktor initialisiert eine Datenstruktur, die in allen Datei-basierten Dialogfenstern benutzt wird (beispielsweise Dialogfenster, die Dateien öffnen oder sichern).

```
⋮
//////////////////////////////////////////////////////////
// DrawApp
// =====
//
DrawApp::DrawApp () : TApplication("Sample ObjectWindows Program")
{
   // Allgemeine Datei-Flags und Filter für Öffnen/Speichern
   // unter-Dialoge.
   // Dateiname und Verzeichnis wird in den Elementfunktionen
   // CmFileOpen und CmFileSaveAs erzeugt.
```

```
     FileData.Flags = OFN_FILEMUSTEXIST | OFN_HIDEREADONLY |
                      OFN_OVERWRITEPROMPT;
     FileData.SetFilter ("AllFiles (*.*)|*.*|");
     // EINFUEGEN<< Fuegen Sie hier Ihren Konstruktor-Code ein.
  }

  :
```

Beachten Sie den Kommentar bezüglich der Plazierung des Codes, der vom Benutzer hinzugefügt wird. AppExpert plaziert ähnliche Kommentare in viele der von ihm definierten Funktionen. Im allgemeinen sollten Sie Code diesen Kommentarpositionen hinzufügen, um die Funktionalität Ihrer Anwendungen zu vervollständigen. Wenn nötig, können Sie Code außerhalb dieser vorgesehenen Bereiche einfügen; auf jeden Fall sollten Sie es vermeiden, irgendeinen der von IDE Experts erzeugten Codes zu ändern.

Der Konstruktor von DrawApp ist nur ein Platzhalter. Der nächste interessante Code ist die Definition der Funktion *InitMainWindow*. In erster Linie instantiiert diese Funktion die Klasse *SDIDecFrame* und erklärt die Instanz zum Hauptfenster der Anwendung. Der Aufruf des Konstruktors für *SDIDecFrame* übergibt Null als Zeiger des Client-Fensters.

```
:
DrawApp::~DrawApp ()
  {
    // EINFUEGEN>> Fuegen Sie hier Ihren Destruktor-Code ein.
  }
  ////////////////////////////////////////////////////////////
  // DrawApp
  // =====
  // Initialisierung der Anwendung.
  //
  void DrawApp::InitMainWindow ()
  {
     SDIDecFrame *frame = new SDIDecFrame(0, GetName(), 0, false);
     nCmdShow = nCmdShow != SW_SHOWMINIMIZED ? SW_SHOWNORMAL :
                                               nCmdShow;

  :
```

Wenn Sie sich den Konstruktor *SDIDecFram*e ansehen, werden Sie feststellen, daß er eine Instanz von *DrawWindow* alloziert und diese, wenn ihm keine andere übergeben wird, zu seinem Client-Fenster erklärt. Der Destruktor für *SDIDecFrame* dient auch hier lediglich wieder als Platzhalter.

```
  :
  //{{SDIDecFrame Implementierung}}
  SDIDecFrame::SDIDecFrame (TWindow *parent, const char far *title,
                            TWindow *clientWnd,
```

```
        bool trackMenuSelection, TModule *module)
          : TDecoratedFrame(parent, title, clientWnd == 0 ? new
DrawWindow(0, "") : clientWnd,
          trackMenuSelection, module)
        {
          // EINFUEGEN>> Fuegen Sie hier Ihren Konstruktor-Code ein.
        }
        SDIDecFrame::~SDIDecFrame ()
        {
          // EINFUEGEN>> Fuegen Sie hier Ihren Destruktor-Code ein.
        }
        .
        .
        .
```

Client-Fenster

Die von AppExpert erzeugte Fensterklasse ist *DrawWindow*. Sie ist definiert in drawwndw.h und DRAWWNDW.CPP. Wenn Sie sich diesen Code ansehen, werden Sie feststellen, daß AppExpert nur einen Konstruktor und einen Destruktor generiert. Der Konstruktor führt eine einzige Funktion aus: Er ruft den Konstruktor für die Basisklasse *TWindow* auf. Der Destruktor von *DrawWindow* ruft als Vorsichtsmaßnahme *Destroy* auf. Hierdurch wird sichergestellt, daß Ihre Anwendung ein sauberes Beenden Ihres Programms ermöglicht.

Überblick über die Optionen von AppExpert

Dieser Abschnitt erläutert die Optionen, die über das AppExpert-Dialogfenster *Optionen für die Anwendungserzeugung* verfügbar sind. Themen und Optionen sind in der Reihenfolge beschrieben, in der sie in der Themenliste erscheinen.

Anwendung

Die hier aufgelisteten Anwendungsoptionen steuern das allgemeine Erscheinungsbild und die Handhabe Ihrer Anwendung.

MDI (Multiple Document Interface) setzt den Typ Ihrer Anwendung auf das Multi-Dokumentschnittstellen-Modell, womit Sie mehrere Objekte zur gleichen Zeit öffnen können. In diesem Modell kann ein einzelnes Objekt gleichzeitig in verschiedenen Ansichten gezeigt werden, unterschiedliche Objekte können durch die gleiche View-Klasse angezeigt werden oder unterschiedliche Objekte können durch unterschiedliche Ansichten gezeigt werden. Die IDE von Borland C++ ist ein Beispiel für eine MDI-Anwendung.

SDI (Single Document Interface) setzt den Typ Ihrer Anwendung auf das Einzel-Dokumentschnittstellen-Modell. Dieses Modell läßt Sie nur ein Objekt zur gleichen Zeit öffnen. Wenn Sie ein neues Objekt öffnen

und ein anderes bereits geöffnet ist, versucht der Dokument-Manager, das erste Objekt zu schließen, bevor er das neue Objekt öffnet.

Dialog Client erzeugt eine Anwendung, die ein Dialogfenster als Haupt-Benutzerschnittstelle verwendet. In diesem Modell ist das Client-Fenster ein Dialogfeld. Der Turbo-Debugger TDWINI.EXE ist ein Beispiel für eine solche Dialog Client-Anwendung.

Dok/View legt fest, ob Ihre Anwendung das Dok/View-Modell für die Behandlung von Anwendungsobjekten unterstützt. Als*Dokument* werden die Daten bezeichnet und als *View* die Schnittstelle zwischen Benutzer und Daten. Im Dok/View-Modell wird zwischen Dokument und View unterschieden (weitere Informationen über Dok/View finden Sie im *ObjectWindows Programmierhandbuch*). Sie können diese Option sowohl für SDI- als auch für MDI-und Dialog Client-Anwendungen verwenden.

Mauspalette erzeugt eine Mauspalette am oberen Rand des Hauptfensters Ihrer Anwendung.

Statuszeile plaziert eine Statuszeile am unteren Rand des Hauptfensters Ihrer Anwendung und erzeugt Code zur Anzeige von Hilfstexten bzw. Hinweisen. Diese werden angezeigt, sobald der zugehörige Menüpunkt markiert wird.

Drag/drop unterstützt Windows-Standard-Aktionen zum Ziehen und Ablegen (Drag & Drop).

Drucken unterstützt druckbezogene Aktivitäten und erzeugt die Menüpunkte *Druckerinstallation*, *Druckvorschau* und *Drucken* im Menü *Datei*.

Grundoptionen

Über die Grundoptionen definieren Sie, wo der generierte Code abgelegt wird, ob Online-Hilfe zu Verfügung gestellt werden soll, welchen Startzustand Ihre Anwendung haben soll und welche Art von Steuerelementen Sie verwenden wollen.

Name der Zieldatei definiert den Namen des zu erstellenden Projekts als Grundlage für die Namensbildung der anderen Elemente dieses Projekts (die Namen für die Header-Dateien, Klassendatenbank, Anwendungsklasse und die Quelldateien basieren auf den Namen in diesem Feld)

Basisverzeichnis enthält die Pfadangabe für das Basisverzeichnis Ihres Projekts. Alle Projektpfade beziehen sich auf dieses Verzeichnis. Sie können das Verzeichnis eigenhändig eingeben oder über das Dialogfenster *Durchsuchen* auswählen. Der Name dieses Verzeichnisses wird für die neue AppExpert-Zieldatei an den Projekt-Manager übergeben. Voreinstellung für das Basisverzeichnis ist der Eintrag im Dialogfeld *Neues AppExpert-Projekt*. Wenn Sie eine AppExpert-Zieldatei einem vorhandenen Pro-

jekt hinzufügen, wird der vorgegebene Wert für das Basisverzeichnis dem vorhandenen Projekt entnommen. Wenn Sie ein neues, noch nicht vorhandenes Verzeichnis angeben, dann übernimmt AppExpert die Erstellung dieses Verzeichnisses.

Hilfedatei unterstützen generiert Hilfe-Quelldateien (.RTF) und eine Hilfe-Projektdatei (.HPJ). Die Hilfe-Projektdatei wird in das Projekt des Projekt-Managers eingegliedert und automatisch zusammen mit der Zieldatei erstellt. Die Hilfe-Quelldatei enthält Platzhalter für die einzelnen Menüpunkte der Anwendung.

Name der Hilfedatei benennt die mit der Anwendung verbundenen Hilfedateien (.HLP und .HPJ).

Über die Schaltfelder für den Anwendungs-Startzustand wird der Anfangsstatus des Hauptfensters der Anwendung auf folgende Weise festgelegt:

- **Normal** (Voreinstellung) gibt das Arbeitsfenster beim Start in der Standardgröße aus, welche über WS_NORMAL definiert ist.
- **Als Symbol** gibt das Fenster beim Start als Symbol auf der Arbeitsfläche von Windows aus.
- **Zoomfeld** gibt das Hauptfenster nach dem Start so groß aus, daß es die gesamte Arbeitsfläche von Windows ausfüllt.

Über die Schaltfelder für Dialogelement-Eigenschaften geben Sie an, welche Typen von Dialogelementen in der Anwendung eingesetzt werden:

- **Windows** (Voreinstellung) verwendet die Standarddialogelemente von Windows.
- **BWCC** verwendet die von Borland für Windows definierten Dialogelemente.
- **3D** verwendet die dreidimensionalen Dialogelemente von Windows.

Optionen für OLE 2

Die Optionen für OLE 2 (Object Linking and Embedding) bestimmen den gewünschten Grad der Unterstützung durch OLE 2 für Ihre Anwendung. Verwenden Sie die folgenden OLE 2-Container-Optionen, um festzulegen, ob Sie eine OLE 2-Container-Anwendung erstellen wollen:

Anwendung ist kein OLE 2-Container (Voreinstellung) legt fest, daß Ihre Anwendung kein OLE 2-Container ist.

Anwendung ist OLE 2-Container bestimmt, daß Ihre Anwendung ein OLE 2-Container sein soll.

Verwenden Sie die folgenden OLE 2-Server-Optionen, um den Typ der OLE 2-Server-Anwendung zu bestimmen, die Sie erzeugen wollen:

Anwendung ist kein OLE 2-Server (Voreinstellung) legt fest, daß die Anwendung kein OLE 2 Server ist.

Anwendung ist OLE 2-Server DLL spezifiziert die Anwendung als OLE 2-.DLL-Server.

Anwendung ist OLE 2-Server EXE spezifiziert die Anwendung als OLE 2-.EXE-Server.

Automatisierung zulassen legt fest, daß die Anwendung nun Objekte für die Automatisierung durch OLE 2 freigeben kann. Die besten Ergebnisse erzielen Sie, wenn Sie ClassExpert zum Auswählen der für die Automatisierung freizugebenden Objekte verwenden.

Server-ID erzeugt eine eindeutige OLE 2 GUID- oder USID-Server-Kennziffer. Sie können dieses Feld zwar bearbeiten, doch dürfte es das beste sein, die Kennziffer vom System erzeugen zu lassen.

Code-Generierung

Die Optionen zur Code-Generierung behandeln verschiedene Aspekte des Code-Generierungsprozesses und legen fest, wo der erzeugte Code abgespeichert werden soll.

Zieldateiname enthält den Namen des Projekts, wie er über *Grundoptionen/Zieldateiname* definiert wurde.

Basisverzeichnis enthält die Pfadangabe für das Basisverzeichnis des Projekts, wie sie über *Grundoptionen/Basisverzeichnis* definiert wurde.

Quellverzeichnis gibt das Verzeichnis an, in dem die Quelldateien der Anwendung abgespeichert sind. Dieser Pfad wird relativ zum Basisverzeichnis angegeben (siehe »Grundoptionen«). Wird ein absoluter Pfad angegeben, dann wird dieser in einen Pfad relativ zum Basisverzeichnis umgewandelt (hierbei ist es nicht gestattet, ein anderes Laufwerk anzugeben). Sie können das Verzeichnis eigenhändig eingeben oder ein vorhandenes auswählen (klicken Sie dazu den Schalter *Durchsuchen* an). Standardmäßig lautet die Angabe für das Quellverzeichnis ».\« (es handelt sich um das Basisverzeichnis).

Header-Verzeichnis gibt das Verzeichnis an, in dem die Header-Dateien der Anwendung abgespeichert sind. Dieser Pfad wird relativ zum Basisverzeichnis angegeben. Wird ein absoluter Pfad angegeben, dann wird dieser in einen Pfad relativ zum Basisverzeichnis umgewandelt (hierbei ist es nicht gestattet, ein anderes Laufwerk anzugeben). Standardmäßig lautet die Angabe für das Header-Verzeichnis ».\« (es handelt sich um das Basisverzeichnis).

Haupt-Quelltext bezeichnet die Haupt-Quelldatei der Anwendung.

Haupt-Header bezeichnet die wichtigste Header-Datei der Anwendung.

Anwendungsklasse enthält die Klasse, die von *TApplication* über AppExpert abgeleitet wurde. Standardmäßig beruht die Bezeichnung der Klasse auf dem Projektnamen.

Info-Dialog-Klasse gibt die Klasse an, die von *TDialog* über AppExpert abgeleitet wurde. Standardmäßig beruht die Bezeichnung der Klasse auf dem Projektnamen.

Kommentare dokumentieren den generierten Code partiell (knapp) oder vollständig (ausführlich) dokumentiert.

Verwaltungsoptionen

Verwaltungsoptionen bestimmen allgemeine Informationen über die Anwendung. Einige der Informationen werden in einen Kommentarblock am Anfang der von AppExpert generierten Dateien eingearbeitet. Einige Informationen werden auch in dem zur Anwendung gehörenden Dialogfenster *Hilfe/Info* angezeigt.

Versionsnummer setzt die Versionsnummer des Projekts, die im Dialogfenster *Hilfe/Info* angezeigt wird (Standard-Versionsnummer ist »1.0«). Diese Information wird in der .RC-Datei des Projekts abgelegt.

Copyright definiert die Copyright-Informationen, die später im Dialogfenster *Hilfe/Info* erscheinen.

Beschreibung enthält eine Kurzbeschreibung der Anwendung. Dieser Text erscheint später im Dialogfenster *Hilfe/Info* der Anwendung. Standardmäßig wird bei Beschreibung der Name des Projekts eingetragen.

Autor enthält den Namen des Programmierers, der den Quelltext entwickelt hat. Diese Angabe wird für die Kommentierung des Quelltextes verwendet.

Firma enthält den Namen der Software-Firma des Programmierers. Diese Angabe wird für die Kommentierung des Quelltextes verwendet.

Hauptfenster-Optionen

Über die Hauptfenster-Optionen werden die Leistungsmerkmale des Hauptfensters einer Anwendung – also dessen Erscheinungsbild und Typ – gesteuert.

Fenstertitel enthält den Text, der in der Titelleiste des Hauptfensters erscheinen soll.

Hintergrundfarbe bestimmt die Hintergrundfarbe für das Hauptfenster der Anwendung. Klicken Sie den Schalter *Hintergrundfarbe* an, um eine Farbe auswählen zu können.

Grundoptionen

Die Grundoptionen steuern das allgemeine Erscheinungsbild des Hauptfensters einer Anwendung. Die Markierungsfelder für Fensterstile

steuern das Erscheinungsbild der Nicht-Client-Bereiche des Hauptfensters Ihrer Anwendung. Weitere Informationen finden Sie in der API Online-Hilfe unter dem Stichwort *Fenster erstellen*.

Titel erstellt einen einzelnen, dünnen Rahmen und eine Titelleiste, in der die Überschrift angezeigt werden kann.

Rand erzeugt einen einfachen, dünnen Rahmen ohne Titelleiste rund um das Hauptfenster.

Vollbildfeld fügt ein Vollbildfeld an der rechten Seite der Titelleiste des Hauptfensters ein. Diese Option steht nur dann zur Verfügung, wenn die Option *Titel* gesetzt ist.

Symbolfeld fügt ein Symbolfeld an der rechten Seite der Titelleiste des Hauptfensters ein. Diese Option steht nur dann zur Verfügung, wenn die Option *Titel* gesetzt ist.

Vertikale Bildlaufleiste fügt eine vertikale Bildlaufleiste an der rechten Seite des Hauptfensters ein.

Horizontale Bildlaufleiste fügt eine horizontale Bildlaufleiste am unteren Rand des Hauptfensters ein.

Systemmenü fügt einen Systemmenüschalter auf der linken Seite der Titelleiste des Hauptfensters ein. Diese Option steht nur dann zur Verfügung, wenn die Option *Titel* gesetzt ist.

Sichtbar aktiviert die Anzeige des Hauptfensters bei Programmstart. Ist *Sichtbar* ausgeschaltet, dann wird der Stil WS_VISIBLE in NOT WS_VISIBLE umgewandelt.

Über **Deaktiviert** wird das Hauptfenster standardmäßig deaktiviert (beim Start der Anwendung wird deren Symbol angezeigt).

Dialograhmen erzeugt einen doppelten Rahmen um das Hauptfenster und ermöglicht, daß die Größe des Fensters verändert werden kann.

Schutz (gleichg.) schützt die dem Hauptfenster gleichgestellten Fenster. Dabei wird die Ausgabe auf das Hauptfenster beschränkt (siehe WS_CLIPSIBLINGS in der API Online-Hilfe).

Schutz (unterg.) schützt die untergeordneten Fenster vor dem Überschreiben durch das Hauptfenster der Anwendung (siehe WS_CLIPCHILDREN in der API Online-Hilfe).

SDI Client

Ein SDI Client (Single Document Interface, Einzel-Dokumentschnittstelle) definiert die Klasse, die den Client-Bereich eines SDI-Hauptfensters repräsentiert.

Client/View-Klasse gibt die Klasse des SDI Client-Bereichsfensters oder des Views an. Die Interpretation dieses Wertes hängt davon ab, ob die Option *Dokument/View* in den Modelleinstellungen der Anwendung aktiviert wurde oder nicht. Wenn ja, dann markiert Client/View-Klasse die View-Klasse des *Standarddokument/View*. Wenn nicht, dann markiert Client/View-Klasse die Klasse des Client-Fensters.

Dokument/View EIN	Dokument/View AUS
TEditView (Voreinstellung)	*TEditFile* (Voreinstellung)
TListView	*TListBox*
TWindowView	*TWindow*

Dokumentklasse (standardmäßg *TFileDocument*) gibt den Namen der Klasse des Standarddokuments an (steht zur Verfügung, wenn *Dokument/View* aktiviert ist).

Beschreibung beschreibt die Klasse der Dateien, die mit *Dokument/View* verbunden sind. Standardwert ist hierbei »Alle Dateien (*.*)«.

Mit **Filter** (standardmäßig »*.*«) bestimmen Sie die Dateiangaben für die Namen der Dateien, auf die die Anwendung zugreifen kann. Sie können mehrere Dateien angeben (durch Semikola voneinander getrennt) oder Joker-Zeichen für Dateinamen verwenden. Dieser Wert wird an die allgemeinen Dateidialogfenster von Windows übergeben, damit die herausgefilterten Dateien in diesen angezeigt werden können.

Standarderweiterung gibt die Standard-Dateinamenserweiterung an. Dieser Wert wird an die allgemeinen Datei-Dialogfenster von Windows übergeben und dann bei der Erstellung einer Datei zum Dateinamen hinzugefügt, wenn keine Erweiterung angegeben wurde. Die Standarderweiterung wird in den Dialogfenstern *Datei/Öffnen* und *Datei/Neu* verwendet.

Hinweis Die Einstellungen *Beschreibung*, *Filter* und *Standarderweiterung* betreffen nur die Dialogfelder *Datei/Öffnen* und *Datei/Speichern* der Anwendung.

Klasse ist der Name, den AppExpert für die den Client-Bereich des SDI-Rahmenfensters repräsentierende Klasse vergibt. Standardmäßig benennt AppExpert die Klasse mit <*Zielname*><*Client-Klasse*>. Sie wird aus der Klasse abgeleitet, die im Textfeld *Client/View-Klasse* angegeben ist.

Quelltextdatei gibt den Namen der Quelldatei an, in der die Implementierung der im Textfeld *Client/View-Klasse* angegebenen Klasse abgespeichert ist. Standardmäßig ist der Name dieser Klasse <*Zielname*><*Client-Klasse*>.CPP (wenn nötig, kürzt AppExpert ihn auf acht Zeichen ab).

Header-Datei gibt die Header-Datei an, in der die Definition der im Textfeld *Client/View-Klasse* angegebenen Klasse gespeichert ist. Vorgegeben ist der Dateiname <*Zielname*><*Client-Klasse*>.h (wenn nötig, kürzt AppExpert ihn auf acht Zeichen ab).

MDI Client

Die Einstellungen für den MDI Client (Multiple Document Interface, Multi-Dokumentschnittstelle) definieren die Client-Fensterklasse eines MDI-Fensters. Eine MDI-Anwendung zeigt den überwiegenden Teil ihrer Daten in den Client-Fenstern des MDI-Kinds an; weitere Informationen finden Sie im folgenden Abschnitt, »MDI-Child/View-Optionen«. Diese Optionen stehen nur zur Verfügung, wenn Sie unter dem Thema *Anwendung/Grundoptionen* den Punkt *MDI-Einstellungen* auswählen.

Client-Klasse enthält den Namen, den AppExpert für die Klasse verwendet, die von *TMDIClient* abgeleitet wurde, der den Client-Bereich des MDI-Rahmenfensters repräsentiert. Standardmäßig benennt AppExpert diese Klasse <*Zielname*>MDIClient.

Quelltextdatei gibt den Namen der Quelldatei an, in der die Implementierung der Klasse mit der in Client-Klasse angegebenen Bezeichnung abgespeichert ist. Vorgegeben ist der Name <*Client-Klasse*>.CPP; wenn nötig, kürzt AppExpert ihn auf acht Zeichen.

Header-Datei gibt die Header-Datei an, in der die Definition der Klasse gespeichert ist, die in der Client-Klasse benannt ist. Vorgegeben ist der Dateiname <*Client-Klasse*>.h; wenn nötig, kürzt AppExpert ihn auf acht Zeichen.

Dialog Client

Die Optionen für den Dialog Client beschreiben die Klasse, die das Client-Fenster des Haupt-Dialogfensters Ihrer Anwendung definiert.

Client-Klasse enthält den Namen, den AppExpert für die Klasse verwendet, die von *TMDIClient* abgeleitet wurde, der den Client-Bereich des MDI-Rahmenfensters repräsentiert. Standardmäßig nennt AppExpert diese Klasse <*Zielname*>TDLGClient.

Quelltextdatei gibt den Namen der Quelldatei an, in der die Implementierung der Klasse mit der in Client-Klasse angegebenen Bezeichnung abgespeichert ist. Vorgegeben ist der Name <*Client-Klasse*>tdlg.CPP; wenn nötig, kürzt AppExpert ihn auf acht Zeichen.

Header-Datei gibt die Header-Datei an, in der die Definition dieser Klasse gespeichert ist. Vorgegeben ist der Dateiname <*Client-Klasse*>tdlgc.h; wenn nötig, kürzt AppExpert ihn auf acht Zeichen.

Dialog-ID spezifiziert den Bezeichner des als Hauptfenster Ihrer Anwendung dienenden Dialogfensters.

Menüleiste einfügen bestimmt, ob das Hauptfenster Ihrer Anwendung eine Menüleiste erhalten soll.

MDI Child/View-Optionen

MDI Child/View-Optionen definieren die Klasse für untergeordnete Fenster oder für *Dokument/View* (stehen zur Verfügung, wenn die Optionen *MDI* und *Dokument/View* in den Modelleinstellungen der Anwendung aktiviert wurden).

MDI-Kindklasse enthält den Namen der Klasse, die von *TMDIChild* abgeleitet wurde und die den Rahmen der untergeordneten MDI-Standard-Fenster repräsentiert. Der vorgegebene Name ist <*Zielname*>MDIChild.

Quelltextdatei enthält den Namen der Quelldatei, in der die Implementierung der Klasse abgespeichert ist, die den in der untergeordneten MDI-Klasse angegebenen Namen trägt. Vorgabewert ist <*MDI-Kind*>1.CPP. Der Name wird, wenn nötig, auf acht Zeichen gekürzt.

Header-Datei benennt die Header-Datei, in der die Definition dieser Klasse abgespeichert ist. Vorgabewert ist <*MDI-Kind*>1.h. Der Name wird, wenn nötig, auf acht Zeichen gekürzt.

Grundoptionen

Grundoptionen definieren das untergeordnete MDI-Standard-Fenster.

MDI-Client/View-Klasse enthält den Namen der Klasse des MDI-Standard- Views. Die Interpretation dieses Wertes hängt davon ab, ob die Option *Dokument/View* in den Einstellungen der Anwendung aktiviert wurde oder nicht:

Dok/View EIN	Dok/View AUS
TEditView (Voreinstellung)	*TEditFile* (Voreinstellung)
TListView	*TListBox*
TWindowView	*TWindow*

Dokumentklasse (steht nur zur Verfügung, wenn Ihre Anwendung *Dokument/View* unterstützt) enthält den Namen der Dokument-Klasse im *Standard-Dokument/View* (standardmäßig *TFileDocument*).

Beschreibung beschreibt die Klasse der Dateien, die mit dem *Dokument/View* verbunden sind. Der Standardwert lautet »Alle Dateien (*.*)«.

Filter (standardmäßig »*.*«) führt alle Dateiangaben mit Wildcards (Platzhaltern) auf, die die Namen der Dateien angeben, auf die die Anwendung zugreifen kann. Die einzelnen Angaben sind durch Semikola oder Kommata voneinander getrennt. Dieser Wert wird an die allgemeinen Datei-

dialogfenster von Windows übergeben, damit die herausgefilterten Dateien in diesen angezeigt werden können.

Standarderweiterung gibt die Standard-Dateinamenserweiterung an. Dieser Wert wird an die allgemeinen Dateidialogfenster von Windows übergeben und wird dann bei der Erstellung einer Datei zum Dateinamen hinzugefügt, wenn keine Erweiterung angegeben wurde.

Hinweis Die Einstellungen *Beschreibung*, *Filter* und *Standarderweiterung* betreffen nur die Dialogfelder *Datei/Öffnen* und *Datei/Speichern* der Anwendung.

Klasse ist der Name, den AppExpert für die den Client-Bereich des MDI-Rahmenfensters repräsentierende Klasse vergibt. Standardmäßig ist der Name dieser Klasse <*Zielname*><*Client-Klasse*>. Sie wird aus der Klasse abgeleitet, die im MDI-Textfeld *Client/View-Klasse* angegeben ist.

Quelltext-Datei gibt den Namen der Quelldatei an, in der die Implementierung der im MDI-Textfeld *Client/View-Klasse* angegebenen Klasse abgespeichert ist. Der Vorgabewert ist <*Zielname*><*Client-Klasse*>.CPP; wenn nötig, kürzt AppExpert ihn auf acht Zeichen.

Header-Datei gibt die Header-Datei an, in der die Definition der im MDI-Textfeld *Client/View-Klasse* angegebenen Klasse gespeichert ist. Vorgegeben ist der Dateiname <*Zielname*><*Client-Klasse*>.h; wenn nötig, kürzt AppExpert ihn auf acht Zeichen.

K A P I T E L 5

Anwendungen verändern mit ClassExpert

ClassExpert arbeitet mit den Anwendungen, die Sie mit AppExpert erstellt haben. Mit ClassExpert können Sie neue Klassen erzeugen, die Implementierung der Klassen bearbeiten und verfeinern und sich durch den Quelltext von vorhandenen Klassen bewegen. Sie können ClassExpert auch mit Resource Workshop verwenden, um Klassen mit Ressourcen zu verbinden (beispielsweise können Sie eine Klasse von *TDialog* mit einer Dialogressource verbinden). Außerdem zeigt ClassExpert virtuelle Funktionen und Ereignisse für vorhandene Klassen an und prüft die virtuellen Funktionen, die in Ihrem Programm implementiert sind.

ClassExpert starten

Um ClassExpert zu starten, wählen Sie *Projekt/Projekt öffnen*, dann öffnen Sie ein von AppExpert erzeugtes Projekt. Dann klicken Sie zweimal den Zielknoten Ihres Projekts an (ClassExpert ist die Standardansicht für AppExpert Ziele), oder Sie wählen *Ansicht/ClassExpert*. ClassExpert erscheint und listet die Klassen und ihre Implementierung für Ihre Anwendung auf.

Abbildung 5.1: ClassExpert



Grundlagen zu ClassExpert

Das ClassExpert-Fenster besteht aus drei voneinander unabhängigen Ausschnitten: Dem Klassenausschnitt, dem Ereignisausschnitt und dem Bearbeitungsausschnitt. Sie können die Größen der einzelnen Ausschnitte selbst bestimmen, indem Sie ihre Rahmen ziehen. Verändern Sie das ClassExpert-Fenster, so behalten die Ausschnitte ihre relativen Proportionen.

Klassenausschnitt

Der Klassenausschnitt listet die Menge der Klassen auf, die ClassExpert für das aktuelle ausgewählte AppExpert-Ziel im Projekt-Manager verwaltet. ClassExpert verwaltet alle Klassen, die von AppExpert erzeugt wurden und jede Klasse, die Sie mit ClassExpert hinzugefügt haben. Die angezeigte Information in den Ereignis- und Bearbeitungsausschnitten ist abhängig von der Klasse, die Sie im Klassenausschnitt ausgesucht haben. Um den Quelltext des Konstruktors der Klasse im Bearbeitungsausschnitt einzusehen, klicken Sie zweimal eine Klasse im Klassenausschnitt an.

Benutzen Sie das lokale Klassenmenü (hierfür klicken Sie den Klassenausschnitt mit der rechten Maustaste an), um Klassen hinzuzufügen, eine Klasse zu automatisieren, Klassenoptionen zu sehen, Dokumentklassen mit Klassenansichten zu verbinden, den Klassenquelltext oder die Header-Datei einzusehen oder zu verändern und um Resource Workshop zu starten (durch auswählen von *Dialog bearbeiten* oder *Menü bearbeiten*). Einzelheiten über die Befehle des lokalen Klassenmenüs können Sie der Online-Hilfe entnehmen.

Ereignisausschnitt

Der Ereignisausschnitt listet Windows-Meldungen, -Befehle und -Elementbenachrichtigungen, virtuelle Funktionen und Automatisierungen der aus dem Klassenausschnitt ausgewählten Klasse auf. Die aktuelle Liste hängt von der Ableitung (die Basisklasse) der gewählten Klasse ab.

Verwenden Sie das lokale Ereignismenü (klicken Sie mit der rechten Maustaste den Ereignisausschnitt an), um Ereignisbehandlungsroutinen (Event-Handler), virtuelle Funktionen und Instanzvariablen hinzuzufügen oder zu löschen. Sie können Resource Workshop auch mit den Befehlen *Menü bearbeiten* und *Dialog bearbeiten* starten. Einzelheiten über die Befehle des lokalen Ereignismenüs erfahren Sie in der Online-Hilfe.

Bearbeitungsausschnitt

Der Bearbeitungsausschnitt ist ein Editor, der die Quelldatei anzeigt, verknüpft mit der Klasse, die Sie im Klassenausschnitt ausgewählt haben. Der Bearbeitungsausschnitt funktioniert genau wie der IDE-Editor, nur daß Sie im Bearbeitungsausschnitt keine Ausschnitte teilen oder andere Dateien öffnen können. Wenn Sie den IDE-Editor anpassen, wirken sich diese Veränderungen auch auf den Bearbeitungsausschnitt aus.

Die Benutzung des lokalen Bearbeitungsmenüs (Klicken mit der rechten Maustaste) ermöglicht den Zugriff auf häufig verwendete ClassExpert-Be-

fehle wie *Öffnen von Quelltextdateien*, *Durchsehen von Objekten*, *Setzen von Haltepunkten* usw. Ein besonderer Befehl des lokalen Bearbeitungsmenüs lautet *Klasse benutzen* und veranlaßt ClassExpert, den Code zu erzeugen, der eine neue Klasse instantiiert, die Sie Ihrer Anwendung hinzugefügt haben. Weitere Einzelheiten über den lokalen Bearbeitungsausschnitt finden Sie in der Online-Hilfe. Sie können auch mit dem Menü *Bearbeiten* der IDE auf zusätzliche Bearbeitungsbefehle zugreifen.

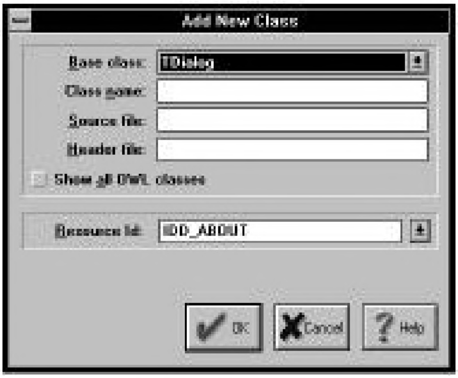
Hinzufügen einer Klasse

ClassExpert ermöglicht das Hinzufügen von ObjectWindows-basierten Klassen und unterstützt eine Vererbungsstufe. Wollen Sie mehrere Ableitungen hinzufügen, müssen Sie dies manuell tun.

Um eine Klasse hinzuzufügen

1. Öffnen Sie das lokale Klassenmenü durch Anklicken des Ausschnitts mit der rechten Maustaste.
2. Wählen Sie *Neue Klasse hinzufügen* oder klicken Sie den entsprechenden Schalter auf der Schalterleiste an. Das Dialogfenster *Neue Klasse hinzufügen* erscheint.

Abbildung 5.2: Dialogfenster *Neue Klasse hinzufügen*



3. Setzen Sie die Optionen in diesem Dialogfenster wie folgt:
 - **Basisklasse** ist die ObjectWindows-Basisklasse, von der Sie Ihre Klasse ableiten wollen.
 - **Klassenname** ist der Name, den Sie der neuen Klasse geben wollen.
 - **Quelldatei** ist der Dateiname, der den Quelltext der Klasse beinhaltet. Die bezeichnete Datei sollte im Quellverzeichnis des Projekts plaziert werden.
 - **Header-Datei** ist der Name der Header-Datei, welche die Klasse definiert. Diese Datei hat den Namen der Quelldatei, verwendet aber die Namenserweiterung .h.

- **Alle OWL-Klassen zeigen** zeigt, wenn angewählt, alle verfügbaren ObjectWindows-Klassen im Kombinationsfenster *Basisklasse* an. Dies erlaubt Ihnen die Auswahl der Nicht-GUI-Klassen, die weder von *TEventHandler* noch von *TWindow* abgeleitet sind.

4. Setzen Sie die folgenden Optionen gemäß der Auswahl Ihrer Basisklasse:

 - **Ressourcen-ID** gibt an, ob Ihr Basisklasseneintrag *TDialog* ist. Benutzen Sie Ressourcen-ID zur Bestimmung der Dialogfenstervorlage, die mit *TDialog* verbunden ist. Das Kombinationsfenster *Ressourcen-ID* enthält Kennungen für alle aktuellen Dialogressourcen Ihrer Anwendung. Wenn Sie eine Kennung bestimmen, die nicht existiert, erzeugt ClassExpert eine leere Dialogfenstervorlage mit dem Bezeichner, den Sie bestimmen (aus Gründen der Einheitlichkeit verwenden Sie das Präfix IDD_). ClassExpert lädt dann Resource Workshop, so daß Sie das Dialogfenster definieren können.
 - **Class Client** zeigt an, ob Ihre Basisklassenauswahl *TFrameWindow* oder eine von *TFrameWindow* abgeleitete Klasse ist. Wählen Sie eine bestehende Klasse aus dem Kombinationsfenster *Client Class*, um den Client-Bereich des neuen Rahmenfensters darzustellen. Beachten Sie, daß dies der einzige Wert ist, den Sie nach Erzeugen der Klasse ändern können.
 - **Fenstereigenschaften bestimmen** zeigt an, ob die Basisklasse *TWindow* oder eine von *TWindow* abgeleitete Klasse ist. Wenn Sie die Schaltfläche *Fenstereigenschaften bestimmen* wählen, erscheint das Dialogfenster *Fensterart*, mit dem Sie die Fenstereigenschaften (wie die Fensterfarbe, den Rahmen, den Titel und so weiter) des neuen Rahmenfensters festlegen können. Zu weiteren Informationen über die einzelnen Fensteroptionen lesen Sie bitte den Abschnitt »Hauptfenster«. Wählen Sie OK, wenn Sie fertig sind.

5. Wählen Sie erneut OK, um die neue Klasse einzufügen, und lassen Sie ClassExpert einen Quellknoten zum Projektbaum anhängen.

Erstellen von Dokumenttypen

Wenn Sie eine Anwendung erstellen, die Dok/View unterstützt, können Sie mit ClassExpert Ansichtsklassen und Dokumenttypen erzeugen.

Um einen Dokumenttyp zu erzeugen

1. Mit AppExpert können Sie eine Klasse für die View erzeugen (sofern Sie nicht eine der drei vorgegebenen View-Klassen *TEditView*, *TListView* oder *TWindowView* verwenden wollen).

2. In ClassExpert klicken Sie mit der rechten Maustaste den Klassenausschnitt an, dann wählen Sie aus dem lokalen Menü *Dokumenttypenansicht*.

3. Markieren Sie eine View-Klasse (wenn Sie Ihre eigene Klasse erstellt haben, erscheint sie in dieser Liste). Die Standard-View-Klassen sind:
 - *TEditView*, stellt ein View-Rahmenobjekt (View Wrapper) für die Textbearbeitungsklasse von ObjectWindows zur Verfügung;
 - *TListView*, stellt Views für Listenfelder zur Verfügung;
 - *TWindowView*, stellt fensterbasierte Views zur Verfügung.
4. Das Textfeld *Beschreibung* beschreibt die Klasse der Dateien, die mit den Listen der Typdateien verbunden ist, die in standardmäßigen Dateidialogfenstern von Windows erscheinen. Der Standardwert ist `Alle Dateien (*.*)`. Der Wert, den Sie angeben, dient zur Beschreibung des Wertes, der in die Filter eingegeben wird.
5. Benutzen Sie *Filter*, um Joker-Zeichen für die Dateibezeichnung aufzulisten (unterschiedliche Filter werden durch Kommata getrennt). Die Filter, die Sie beschrieben haben, werden an die standardmäßigen Dateidialogfenster von Windows weitergegeben, um die in diesen Dialogfenstern erscheinenden Dateien zu filtern. Wenn Sie beispielsweise einen Dokumenttyp für Bitmaps eingeben, können Sie den Filter `*.BMP` haben. Der Standardwert für dieses Feld ist *.*, und der in *Beschreibung* eingegebene Wert dient als Erklärung dieses Werts.
6. Mit *Standarddateinamenserweiterung* wird die vorgegebene Erweiterung des Dateinamens bestimmt. Dieser Wert wird an Windows-übliche Dateidialogfenster weitergegeben und denjenigen Dateinamen hinzugefügt, die keine Dateinamenserweiterungen haben. Der Standardwert ist `TXT`.
7. Drücken Sie den Layout-Schalter, um die Eigenschaften für Dok/View festzulegen. Folgende Eigenschaften können festgesetzt werden (weitere Informationen entnehmen Sie bitte der ObjectWindows-Dokumentation):
 - *dtAutoDelete* löscht das Dokumentobjekt, wenn die letzte View geschlossen ist.
 - *dtNoAutoView* erstellt nicht automatisch einen vorgegebenen View-Typ.
 - *dtSingleView* stellt für jedes Dokument eine eigene View zur Verfügung.
 - *dtAutoOpen* öffnet ein Dokument, wenn es erzeugt wird.
 - *dtUpdateDir* aktualisiert das Verzeichnis mit dem Dialogverzeichnis.
 - *dtHidden* blendet die Vorlage von der Auswahlliste der Benutzer aus.
 - *dtSelected* bezeichnet die zuletzt gewählte Vorlage.
 - *dtReadOnly* kennzeichnet das Markierungsfeld *Nur Lesen* nach Erstellung des Dialogfensters.

- *dtOverWritePrompt* fragt den Benutzer, ob er eine existierende Datei wirklich überschreiben will, wenn er das Dialogfenster *Speichern unter* benutzt.
- *dtHideReadOnly* blendet das read-only-Markierungsfeld aus.
- *dtPathMustExist* läßt den Benutzer nur existierende Pfade eingeben.
- *dtFileMustExist* läßt den Benutzer nur existierende Dateinamen eingeben.
- *dtCreatePrompt* fordert vor Erstellung eines neuen Dokuments die Bestätigung des Benutzers.
- *dtNoReadOnly* ermittelt, ob die angegebene Datei beschreibbar ist.

8. Drücken Sie *Hinzufügen*, um den Dokumenttypen ihrer Anwendung hinzuzufügen (dies aktualisiert eine Datenstruktur in der Hauptquelldatei, die alle verfügbaren Dokumenttypen beschreibt).
 Dokument/Ansicht erscheint in der Liste der existierenden Typen.
9. Wiederholen Sie die Schritte 1-8 für jeden zusätzlichen Dokumenttyp, den Sie hinzufügen wollen. Wenn Sie fertig sind, wählen Sie *Schließen*, um zu ClassExpert zurückzukehren.

Hinzufügen und Löschen von Ereignisbehandlungsroutinen (Event-Handlern)

Um eine Ereignisbehandlungsroutine hinzuzufügen, führen Sie die folgenden Schritte aus:

1. Wählen Sie im Klassenausschnitt die Klasse für die Ereignisbehandlungsroutine. Die Ereignisse erscheinen im Ereignisausschnitt.
2. Wählen Sie das Ereignis, das Sie bearbeiten wollen (eventuell müssen Sie die Ereignisliste erweitern), dann klicken Sie das Ereignis mit der rechten Maustatse an, um das lokale Menü einzusehen.
3. Wählen Sie aus dem lokalen Menü *Behandlungsroutine hinzufügen*.
 - Wenn Sie eine Behandlungsroutine für eine Windows-Meldung hinzufügen, fügt ClassExpert der Kommandotabelle einen Eintrag hinzu, dessen Name standardmäßig vorgegeben ist. Die mit der Behandlungsroutine verknüpfte Funktion erscheint im Bearbeitungsausschnitt, definiert mit dem entsprechenden Standardbearbeitungs-Code.
 - Wenn Sie eine Behandlungsroutine für eine virtuelle Funktion hinzufügen, erscheint die mit der Behandlungsroutine verbundene Funktion im Bearbeitungsausschnitt, definiert mit dem entsprechenden Standardbearbeitungs-Code.
 - Wenn Sie eine Behandlungsroutine für einen Befehl oder eine Elementbenachrichtigung hinzufügen, fordert ClassExpert Sie zur Eingabe des Funktionsnamens auf, ehe der Eintrag der Kommandotabelle hinzugefügt wird.

Hinweis

Der Befehl *Behandlungsroutine hinzufügen* des lokalen Menüs ist nur bei Ereignissen anwendbar, die noch nicht von einer Elementfunktion der ausgesuchten Klasse bearbeitet werden. Unbearbeitete Ereignisse und nicht überschriebene virtuelle Funktionen sind im Ereignisausschnitt mit einer nebenstehenden Check-Marke markiert (der Ereignisausschnitt kennzeichnet behandelte Ereignisse mit einer Check-Marke). Eine hellgraue Check-Marke bedeutet, daß *einige* Ereignisse innerhalb der Ereigniskategorie bearbeitet werden (zur Ansicht aller aktuellen Ereignisse erweiteren Sie die Liste).

Um eine Ereignisbehandlungsroutine zu löschen, führen Sie die folgenden Schritte aus:

1. Wäh len Sie die Klasse der Meldungsbehandlungsroutine (Message-Handler) im Klassenauschnitt. Das Ereignis erscheint im Ereignisausschnitt.
2. Wählen Sie das mit der Check-Marke gekennzeichnete Ereignis mit der Behandlungsroutine, das Sie löschen wollen (möglicherweise ist eine Erweiterung der Ereignisliste erforderlich) aus, dann klicken Sie das Ereignis mit der rechten Maustaste an, um das lokale Menü anzusehen.
3. Wählen Sie *Behandlungsroutine löschen*. ClassExpert entfernt die Behandlungsroutine aus dem Ereignisausschnitt, aber der Code zur Implementierung der Funktion wird nicht gelöscht. ClassExpert benachrichtigt Sie darüber, und Sie müssen den Funktions-Code manuell entfernen. Der Code für die Behandlungsroutine erscheint im Ereignisausschnitt, so daß Sie ihn löschen können. Wenn Sie die Funktion löschen, entfernen Sie die Funktionsdefinition aus der Header-Datei (Sie können den Bearbeitungs-Header aus dem lokalen Menü des Klassenausschnitts wählen, um die Datei einzusehen). Wenn Sie ein Ereignis, einen Befehl oder eine Elementbenachrichtigung löschen, entfernt ClassExpert den damit verbundenen Eintrag in der Kommandotabelle.

Hinzufügen und Löschen von Instanzvariablen

Instanzvariablen ermöglichen die unkomplizierte Bearbeitung vieler Elemente. Wenn Sie Instanzvariablen erzeugen, fügt ClassExpert Ihrem Code einen Transferpuffer hinzu. Dieser Transferpuffer sammelt Informationen zur Laufzeit, so daß Sie die Informationen im Transferpuffer benutzen können, anstatt einen Code zu erzeugen, der überprüft, ob jedes Markierungsfeld aktiviert ist. Wenn Sie beispielsweise ein Dialogfenster mit sechs Markierungsfeldelementen haben, und Ihre Anwendung soll etwas ausführen, was auf den markierten Feldern basiert, so können Sie für jedes Element Instanzvariablen benutzen, und dann die Daten des Transferpuffers für Ihren Code verwenden. Weitere Informationen über Transferpuffer und Instanzvariablen entnehmen Sie bitte dem *ObjectWindows Programmierhandbuch*.

Um eine Instanzvariable einem Element hinzuzufügen (beide miteinander zu verbinden), führen Sie die folgenden Schritte aus:

1. Klicken Sie mit der rechten Maustatse *Elementbenachrichtigung* im Ereignisausschnitt an, dann wählen Sie aus dem lokalen Menü *Instanzvariable hinzufügen* (um das Element, das Sie verändern möchten anzusehen , müssen Sie eventuell die Liste der Elementbenachrichtigungen erweitern),
2. Geben Sie im Dialogfenster *Instanzvariable hinzufügen* einen Namen für jede Instanzvariable ein. ClassExpert fügt Ihrem Anwendungs-Code folgendes hinzu:
 - In der Header-Datei wird eine Strukturdeklaration mit einem Eintrag für die Instanzvariable hinzugefügt.
 - In der .CPP-Datei wird die Instanzvariable dem Klassenkonstruktor zugewiesen (dies verbindet die ObjectWindows-Klasse mit dem Ressourcen-Objekt).
 - In der .CPP-Datei wird eine statische Instanz der Transferstruktur deklariert.
3. Wählen Sie OK. *Elementbenachrichtigung* im Ereignisausschnitt zeigt die Klasse und den Namen der von Ihnen erzeugten Instanzvariablen an.

Um eine Instanzvariable zu löschen, führen Sie die folgenden Schritte aus:

1. Wählen Sie das Element mit der Instanzvariablen, die Sie löschen wollen.
2. Klicken Sie das Element mit der rechten Maustaste an und wählen Sie *Instanzvariable löschen* aus dem lokalen Menü,
3. ClassExpert entfernt folgendes aus Ihrem Code:
 - Den Struktureintrag.
 - Die Zeigervariable in der Klassendeklaration.
 - Die Zuweisung der Klassenvariablen, die mit dem Ressourcen-Element im Konstruktor verbunden ist.

Wenn Sie alle Instanzvariablen von Ihrem Code löschen, bleibt eine leere Struktur und der Aufruf zum Setzen des Transferpuffers zurück. Da sich diese Informationen nicht auf den Rest Ihres Codes auswirken, brauchen Sie sie nicht manuell zu entfernen.

Zum Quelltext einer Klasse wechseln

Für die Ansicht des Quelltextes einer Klasse wählen Sie die Klasse im Klassenausschnitt (Klassennamen einmal anklicken). Der Code erscheint im Ereignisausschnitt. Wenn Sie die Einfügeposition im Ereignisausschnitt verschieben, erinnert sich ClassExpert an die Position, wenn Sie das nächste Mal die Klasse auswählen.

Um die Einfügeposition zum Code des Klassenkonstruktors zu bewegen, klicken Sie zweimal den Klassennamen im Klassenausschnitt an. Um zu einem behandelten Ereignis zu wechseln, klicken Sie das Ereignis zweimal mit der Maustaste im Ereignisausschnitt an. Sie können sich die Quelldatei oder ihre Header-Datei auch in einem IDE Editor einsehen:

1. Klicken Sie im Klassenausschnitt zweimal eine Klasse an, um das lokale Menü zu öffnen.
2. Wählen Sie *Quelle bearbeiten*, um die Quell-.CPP-Datei des Klassenkonstruktors einzusehen, oder wählen Sie *Header bearbeiten*, um die Header-Datei, in der die Klasse definiert ist, einzusehen.

Die Verwendung von Resource Workshop unter ClassExpert

Resource Workshop ist der Standard-Viewer für Ressourcenskripte (.RC-Dateien). Wenn Sie Resource Workshop von ClassExpert aus starten (indem Sie *Dialog bearbeiten* oder *Menü Bearbeiten* aus dem lokalen Menü wählen), so lädt Resource Workshop das Ressourcenskript für diese Anwendung automatisch.

Bei der Benutzung von Resource Workshop mit einem durch AppExpert erzeugten Code sollten Sie immer von ClassExpert aus starten, denn Resource Workshop und ClassExpert aktualisieren sich gegenseitig, wenn Sie Veränderungen im Projekt vornehmen. Beim Starten von Resource Workshop überprüft er den Ressourcen-Code auf Änderungen und schickt die Aktualisierungen sofort an ClassExpert. Wenn Sie beispielsweise einem Dialogfenster einen Schalter hinzufügen, teilt Resource Workshop dies ClassExpert mit. ClassExpert fügt dann dieses Element dem Ereignisausschnitt hinzu. Um das Element in ClassExpert einsehen zu können, wählen Sie das Element in Resource Workshop, klicken es mit der rechten Maustaste an und wählen dann ClassExpert aus dem lokalen Menü. Resource Workshop bringt Sie mit dem hervorgehobenen Element im Ereignisausschnitt zu ClassExpert zurück.

Starten von Resource Workshop aus der IDE

Beim Starten von Resource Workshop als Betrachter einer Anwendung (entweder aus dem Projekt-Manager in der IDE oder aus ClassExpert), verhält es sich anders, als wenn Sie Resource Workshop unabhängig starten:

- Wenn Sie Änderungen in Resource Workshop vornehmen, die sich auf die Klassenstruktur oder die Funktionalität auswirken (wie die Bearbeitung eines Menüs oder Dialogfensters), werden diese Änderungen sofort im Fenster aktualisiert.
- Sie können kein anderes Programm öffnen (Kein *Datei/Öffnen* oder *Datei/Neu* möglich).

- Wenn Sie IDE schließen, wird Resource Workshop ebenfalls geschlossen, und jede von Ihnen vorgenommene Änderung wird automatisch gespeichert.
- Wenn Sie die Projektdatei schließen, die Resource Workshop gestartet hat, wird Resource Workshop ebenfalls geschlossen.
- Wenn Sie ein Projekt anlegen, während Resource Workshop geöffnet ist, erzeugt es eine .RES-Datei, die auf den geladenen Ressourcen basiert. Möchten Sie beispielsweise ein Dialogfenster bearbeiten, haben es aber nicht gesichert, dann wird die .RES-Datei die *nichtgespeicherten* Bearbeitungen wiedergeben.
- Sie können von Resource Workshop mit dem lokalen Menü auf die IDE zugreifen (rechts anklicken) und ClassExpert anwählen.

Verwenden von Rescan

Rescan ist ein besonderes Projekthilfsprogramm, das den Quelltext, der in der .IDE-Datei Ihres AppExpert Projekts aufgelistet ist, untersucht und die Datenbank des Projekts (die .APX-Datei) aktualisiert oder wieder anlegt, gemäß dem, was es im Quelltext vorfindet. Rescan sucht nach besonderen Symbolen im Quelltext, um die .APX-Datenbankdatei wiederherzustellen, und startet dann Resource Workshop, um die Informationen über die Projektressourcen wiederherzustellen. Wenn Rescan erfolgreich durchgeführt wurde, wird die ursprüngliche Datei der Projektdatenbank in *.~AP umbenannt und eine neue Datenbankdatei erzeugt; anderenfalls bleibt die ursprüngliche Datenbank in der Form *.APX.

Sie können Rescan verwenden, um

- eine Klasse zu löschen
- eine Klasse von einer Quelldatei in eine andere zu verschieben
- eine Klasse, Behandlungsroutine, Instanzvariable oder Dialog-ID umzubenennen
- eine Klasse aus einem anderen Projekt zu importieren
- eine verlorengegangene oder beschädigte Datei einer Projektdatenbank (*.APX) wiederherzustellen.

Löschen einer Klasse

Um eine Klasse zu löschen, führen Sie die folgenden Schritte aus:

1. Entfernen Sie den Quellknoten der Klasse aus dem Projekt, indem Sie mit der rechten Maustaste den Knoten im Projekt-Manager anklicken, dann wählen Sie *Knoten löschen*. Wenn die Klasse eine Quelldatei gemeinsam mit anderen Klassen benutzt, löschen Sie den Code, der zu der Klasse gehört, aus der Quelldatei. Dann löschen Sie aus der Quelldatei die Referenzen auf diese Klasse.

2. Klicken Sie das AppExpert Ziel im Projekt-Manager mit der rechten Maustaste an, dann wählen Sie *Extras/Rescan*. Rescan sucht die Quelldateien, die als abhängig vom AppExpert-Ziel aufgelistet sind, ab. Resource Workshop sucht die Ressource-Dateien ab und aktualisiert sie. Wenn Rescan fertig ist, kehren Sie zu der aktualisierten Projektdatei zurück, wo Sie entweder Ihre Anwendung anlegen oder benutzen können. Sie können die gelöschte Klasse dem Projekt hinzufügen, indem Sie die Quelldatei der Klasse als abhängig vom Ziel hinzufügen und dann erneut absuchen.

Verschieben einer Klasse

Um eine Klasse von einer Quelldatei in eine andere zu verschieben, führen Sie die folgenden Schritte aus:

1. Verschieben Sie (ausschneiden und einfügen) den Quelltext der Klasse in die neue Datei. Wenn die neue Datei kein Knoten unter dem Zielknoten im Projekt ist, fügen Sie den Knoten dem Projekt hinzu. Wenn die verschobene Klasse ihre eigene Quelldatei hat, können Sie die leere Quelldatei aus dem Projekt löschen.
2. Klicken Sie mit der rechten Maustaste den Zielknoten im Projekt-Manager an, dann wählen Sie *Extras/Rescan*. Wenn Rescan fertig ist, kehrt es wieder in das Projektfenster der IDE zurück.

Umbenennen eines Elements

Um eine Klasse, Ereignisbehandlungsroutine (Event-Handler), Instanzvariable oder eine Dialog-ID umzubenennen, führen Sie die folgenden Schritte aus:

1. Benutzen Sie den IDE-Editor, um alle Nennungen des ursprünglichen Namens zu suchen und durch den neuen Namen zu ersetzen. Alle mit dem Projekt verbundenen Quelldateien (.CPP, .H, .RC, und .RH-Dateien) müssen markiert sein.
2. Klicken Sie mit der rechten Maustaste den Zielknoten im Projekt-Manager an, dann wählen Sie *Extras/Rescan*. Wenn Rescan durchgeführt wurde, kehrt es wieder zum Projekt-Manager zurück.

Hinweis

Nach der Umbenennung eines Elements überprüfen Sie, ob die von AppExpert erzeugten `#include`, `#ifdef` und `#define` Anweisungen weiterhin gelten. Diese Anweisungen können verändert werden, während ClassExpert die Klassennamen in Ihren Quelldateien sucht oder ersetzt.

Importieren einer Klasse

Um eine Klasse aus einem Projekt in ein anderes zu importieren, führen Sie folgende Schritte aus:

1. Verschieben oder kopieren Sie die Quell- und Header-Datei, die die Klasse des Quell- und Header-Verzeichnisses des neuen Projekts beschreibt. Alle Quelldateien eines Projektes müssen sich im Quellverzeichnis (.CPC-Dateien) oder Header-Verzeichnis (.h-Dateien) des Projektes befinden. Diese Verzeichnisse wurden erzeugt, als Sie zum ersten Mal das AppExpert-Projekt erzeugt haben.
2. Fügen Sie die Quelldatei der Klasse als einen abhängigen Knoten unter dem Zielknoten im IDE-Projekt (verwenden Sie *Knoten hinzufügen* aus dem lokalen Menü) hinzu.
3. Fügen Sie die Header-Datei, in der die Klassendefinition steht, als einen abhängigen Knoten unter der Quellknotendatei im AppExpert-Projekt hinzu (verwenden Sie *Knoten hinzufügen* aus dem lokalen Menü).
4. Klicken Sie mit der rechten Maustaste den AppExpert-Zielknoten im Projektfenster an, wählen Sie dann *Extras/Rescan*.

Wiederanlegen einer .APX-Datenbankdatei

Um eine verlorengegangene oder beschädigte Datenbankdatei (.APX-Datei) wieder anzulegen, führen Sie folgende Schritte aus:

1. Öffnen Sie die Projektdatei, in der das AppExpert-Ziel und abhängige Knoten stehen (dies ist die .IDE-Datei).
2. Klicken Sie mit der rechten Maustaste das AppExpert-Ziel an, dann wählen Sie *Extras/Rescan* aus dem lokalen Menü. Rescan erstellt automatisch eine neue Datenbankdatei, indem es Symbole aus dem Quelltext der Anwendung benutzt.

K A P I T E L 6

Debuggen in der IDE

Ganz gleich, wie sorgfältig Sie beim Programmieren arbeiten, Ihre Programme enthalten sicher zunächst Fehler (Bugs), die verhindern, daß sie so, wie von Ihnen geplant, ablaufen. Unter »Debuggen« verstehen wir die Tätigkeit, Programmfehler, die die korrekte Funktion Ihrer Programme verhindern, zu finden und zu korrigieren. Dieses Kapitel beschreibt, wie Sie Fehler in Ihren Windows-Programmen mit Hilfe der IDE lokalisieren und korrigieren können.

Arten von Fehlern

Es gibt drei Grundtypen von Programmfehlern: solche, die sich bei der Compilierung zeigen, solche, die sich beim Ablauf zeigen, und schließlich logische Fehler.

Compilierzeitfehler

Compilierzeitfehler, auch *Syntax*fehler genannt, entstehen, wenn Ihr Programm gegen eine Syntaxregel von C oder C++ verstößt. Die IDE kann Ihr Programm nicht compilieren, solange es nicht ausschließlich zulässige C- oder C++ Anweisungen enthält.

Wenn Ihr Quelltext Syntaxfehler enthält, öffnet der Compiler das Fenster *Meldungen* und zeigt alle Fehler und Warnungen an. Die Fehler müssen einer nach dem anderen behoben werden. Um einen Fehler zu korrigieren, doppelklicken Sie darauf; die IDE stellt dann den Cursor auf diejenige Zeile im Quelltext, die das Problem verursacht, so daß Sie dort korrigieren können. Wenn Ihr Quellprogramm mehr als einen Syntaxfehler enthält, wiederholen Sie das eben Beschriebene für jeden einzelnen Fehler, bis Sie alle korrigiert haben.

Die im Fenster *Meldungen* angezeigten Warnungen bewirken keinen Stop der Compilierung, aber sie weisen auf Programmstellen hin, die Sie vielleicht auf mögliche Probleme hin überprüfen sollten. Warnungen weisen

Sie beispielsweise darauf hin, daß Ihr Programm nicht portabel ist, daß es nicht dem ANSI-Standard entspricht oder daß es zu ineffektiven Maschinenbefehlen führt. Sie können selbst wählen, welche Art von Warnungen Ihnen angezeigt werden sollen:

1. Wählen Sie *Optionen/Projekt* und doppelklicken Sie auf *Meldunge*n.
 Die IDE zeigt dann alle der verschiedenen Meldungskategorien an; auf der rechten Seite erscheinen außerdem die Einstellungen, die diese Meldungen betreffen.

2. Wählen Sie die gewünschte Option. Diese Optionen bestimmen, welche Art von Warnungen die IDE anzeigt.

 - Wählen Sie *Alle*, so zeigt die IDE alle Warnungen an.
 - Wählen Sie *Markierte*, so zeigt die IDE nur die ausgewählten Warnmeldungen an. (Das Auswählen der Meldungen wird unter Punkt 4 beschrieben).
 - Wählen Sie *Keine*, so zeigt die IDE keine Warnmeldungen an.

1. Begrenzen Sie die Anzahl der im Fenster *Meldungen* angezeigten Fehler- und Warnungsmeldungen. Geben Sie dazu in den Feldern *Abbruch nach* die maximale Zahl von Fehler- und Warnungsmeldungen vor, die Sie bei jedem Compilierungs-Lauf angezeigt erhalten möchten. Zulässig ist jede ganze Zahl im Bereich von 0 bis 255.
 Wenn Sie in einem der Felder 0 eingeben, arbeitet die IDE ohne Begrenzung der Anzahl der angezeigten Meldungen im Fenster *Meldungen*.

2. Wählen Sie unter *Themen* die gewünschte Kategorie der Warnmeldungen.

3. Wählen Sie auf der rechten Seite die speziellen Meldungen, die die IDE anzeigen sollen.

Wenn Sie beispielsweise möchten, daß Sie der Compiler bei verdächtigen Zeigerumwandlungen, die möglicherweise dem ANSI-Standard widersprechen, warnt, so wählen Sie *Verstöße gegen ANSI* und aktivieren die Option *Möglicherweise inkorrekte Zeigerumwandlung*.
Häufige Ursache für Compilierzeitfehler sind Tippfehler, fehlende Semikola, Bezüge auf nicht deklarierte Variablen, Funktionsaufrufe mit Parametern falschen Typs oder in falscher Anzahl sowie Zuweisungen von Werten des falschen Typs an Variablen.

Nachdem Sie alle Fehler korrigiert haben, starten Sie die Compilierung nochmals. Nachdem Sie so schließlich alle Syntaxfehler beseitigt haben und sich Ihr Programm fehlerfrei compilieren läßt, können Sie daran gehen, das Programm ablaufen zu lassen und nach Laufzeitfehlern und logischen Fehlern zu suchen.

Laufzeitfehler

Wenn sich Ihr Programm fehlerfrei compilieren läßt, aber abstürzt, wenn Sie versuchen, es ablaufen zu lassen, so haben Sie es mit einem Laufzeitfehler zu tun. Ihr Programm besteht dann zwar aus an sich zulässigen Anweisungen, aber bei deren Ausführung geht etwas schief. Es könnte zum Beispiel sein, daß Ihr Programm versucht, eine nicht vorhandene Datei zu öffnen oder eine Zahl durch null zu dividieren. Das Betriebssystem erkennt solche Situationen und bricht die Ausführung Ihres Programms ab.

Logische Fehler

Logische Fehler sind Fehler im Entwurf und in dessen Umsetzung in die Programmiersprache. Ihre Anweisungen sind zulässig und tun auch *irgend etwas*, aber nicht das von Ihnen Beabsichtigte. Solche Fehler sind oft schwer aufzuspüren, weil die IDE sie nicht automatisch finden kann. Glücklicherweise verfügt die IDE aber über Möglichkeiten zum Debuggen, die Ihnen beim Finden logischer Fehler helfen können.

Logische Fehler entstehen zum Beispiel, wenn Variablen falsche oder unerwartete Werte annehmen, wenn Grafikbilder verfälscht sind oder wenn Programmteile nicht dann durchlaufen werden, wenn Sie es erwarten. Der Rest dieses Kapitels erläutert Techniken zum Auffinden solcher logischen Fehler.

Erzeugen von Debug-Informationen

Sie müssen Ihr Programm so compilieren und linken, daß Debug-Informationen in seine OBJ- und EXE-Dateien eingefügt werden.

- Um Debug-Informationen in Ihre OBJ-Dateien aufzunehmen, wählen Sie *Optionen/Projekt*, um das Dialogfenster *Standard-Projekt-Optionen* zu öffnen, und wählen Sie dann *Compiler/Debugger/Debug-Information in OBJ-Dateien*. Dies ist die Standardeinstellung.
- Um Debug-Informationen in Ihre EXE-Dateien aufzunehmen, wählen Sie im Dialogfenster *Standard-Projekt-Optionen* den Punkt *Linker/Allgemein/Debug-Informationen mit aufnehmen*. Dies ist die Standardeinstellung.

Wenn Sie nun das Programm compilieren, erzeugt der Compiler eine spezielle Tabelle aller verwendeten Namen und schreibt sie in die EXE-Datei. Diese Liste, *Symboltabelle* genannt, wird vom Debugger benutzt, um alle in Ihrem Programm verwendeten Variablen, Konstanten, Typen, Funktionsnamen und Anweisungen zu überwachen.

Kommandozeilenparameter beim Debuggen

Falls das Programm, das Sie debuggen wollen, auf die Übergabe von Kommandozeilenparametern angewiesen ist, müssen Sie diese wie folgt angeben:

1. Wählen Sie *Optionen/IDE*
2. Geben Sie im Dialogfenster *Aufrufargumente* die Parameter ein, die Sie an das Programm übergeben haben möchten.

Steuerung des Programmablaufs

Das Wichtigste beim Debuggen ist die Steuerung der Ausführung Ihres Programms. Sie können bestimmen, wann jede Anweisung ausgeführt wird, dadurch wird es leichter, herauszufinden, in welchem Teil Ihres Programms das Problem liegt.

Mit *Funktion überspringen* und *Nächste Anweisung* können Sie Ihr Programm Anweisung für Anweisung einzeln ablaufen lassen; die Folgeanweisung wird erst ausgeführt, wenn Sie dem Debugger dies befehlen. Gehen Sie einfach mit *Funktion überspringen* oder *Nächste Anweisung* durch Ihr Quellprogramm, bis Sie den Punkt erreichen, an dem etwas schiefläuft. Dort können Sie dann den Zustand des Programms und seiner Daten überprüfen, sich die Ausgabe des Programms und die Werte seiner Variablen ansehen und schließlich sogar ganze Ausdrücke verändern oder nachrechnen - dies alles, bevor Sie den Debugger die nächste Zeile ausführen lassen.

Beobachten der Programmausgaben

Während Sie mit *Funktion überspringen* oder *Nächste Anweisung* durch Ihr Anwendungsprogramm laufen, können Sie dessen Ausgaben in seinem Fenster beobachten. Positionieren Sie Ihre Fenster am besten so, daß Ihr Quellprogramm und Ihr Anwendungsfenster während des Testlaufs gleichzeitig sichtbar sind. Wenn sich das Fenster des IDE-Desktops und Ihr Anwendungsfenster beim Debuggen überlappen, werden Sie in Ihrem Anwendungsfenster ein leichtes Flackern bemerken. Wenn Sie diese Fenster so anordnen, daß sie sich nicht überlappen, läuft Ihr Programm schneller und »ruckfreier« ab.

Programm mit *Funktion überspringen* ausführen lassen

Jede Programmausführung unter dem Debugger, einschließlich *Funktion überspringen*, *Nächste Anweisung* und Anhalten an Haltepunkten, findet in Schritten von Quelltextzeile zu Quelltextzeile statt. Auch wenn eine Anweisung auf dem Bildschirm im mehrere Zeilen umbrochen wird, zählt sie trotzdem als eine Zeile.

Sie können beim Debuggen nur bis herab zu einer einzigen Quelltextzeile heruntergehen. Wenn Sie also mehrere Anweisungen in eine Zeile geschrieben haben, können Sie diese Anweisungen nicht einzeln debuggen. Andererseits können Sie eine einzige Anweisung für Debug-Zwecke zwar auf mehrere Zeilen verteilen, aber sie wird trotzdem noch als ein einzelner Schritt ausgeführt.

Jedesmal, wenn Sie den Debugger anweisen, einen oder mehrere Schritte auszuführen, bewegt sich der *Ausführungsbalken*, das ist die farblich hervorgehobene Zeile, die die aktuelle Position in Ihrem zu debuggenden Programm markiert, weiter zur Folgezeile. Der Ausführungsbalken zeigt immer auf die Zeile, die als nächste ausgeführt werden wird.



Der Schritt-für-Schritt-Betrieb ist die einfachste Methode, sich stückchenweise durch Ihr Programm zu bewegen. Um die Quelltextzeile, auf der der Ausführungsbalken steht, auszuführen (einschließlich aller dort vor dem Verlassen der Zeile stattfindenden Funktionsaufrufe), wählen Sie entweder *Debug/Funktion überspringen* (oder betätigen Sie die Taste [*F8*] oder klicken Sie auf den Schalter *Funktion überspringen* in der Mauspalette). Der Ausführungsbalken zeigt danach auf die nächste komplette Zeile.

Um zu verstehen, wie *Funktion überspringen* funktioniert, nehmen Sie einmal an, daß die folgenden Zeilen (die ersten eines Beispielprogramms) in einem Editierfenster sichtbar sind:

```
    ⋮

BOOL InitApplication ( HINSTANCE hInstance )
{
   WNDCLASS wc;

   wc.style       = CS_HREDRAW | CS_VREDRAW;
   wc.lpfnWndProc = (long (FAR PASCAL*)())MainWndProc;
   wc.cbClsExtra  = 0;
   wc.cbWndExtra  = 0;
    ⋮

   return ( RegisterClass ( &wc ) );
}
    ⋮

int PASCAL WinMain ( HINSTANCE hInstance, HINSTANCE hPrevInstance,
LPSTR lpCmdLine, int nCmdShow )
{
  MSG msg;

  if ( !hPrevInstance )
  if ( !InitApplication ( hInstance ) )
    return (FALSE );
```

```
    if ( !InitInstance ( hInstance, nCmdShow ) )
      return ( FALSE );
      :
  }
```

In diesem Beispiel ist *InitAnwendung* eine Funktion, die Sie in einem Modul definiert haben, das mit Debug-Information compiliert wurde. Wenn Sie dieses Programm nun wirklich debuggen würden, so würde der Debugger jedesmal, wenn Sie *Debug/Funktion überspringen* wählen (oder [*F8*] betätigen), die durch den Ausführungsbalken hervorgehobene Zeile ausführen, und danach würde sich der Ausführungsbalken auf die Folgezeile bewegen. Wenn Sie *Funktion überspringen* wählen würden, während der Ausführungsbalken gerade auf der Anweisung `if ( !InitAnwendung ( hInstance ) )` steht, so würde der Debugger die Funktion InitAnwendung ausführen (wobei ein boolescher Wert zurückgegeben würde), aber Sie würden den Ausführungszeiger nicht wirklich durch die Funktion *InitAnwendung* laufen sehen. Der Debugger würde die Funktion statt dessen *überspringen*, und Sie würden erst *nach* dem Rücksprung von der Funktion wieder die Kontrolle erhalten.

Programm mit *Nächste Anweisung* ablaufen lassen



Der Ablauf bei *Nächste Anweisung* ist sehr ähnlich dem Ablauf bei *Funktion überspringen*. Der Unterschied besteht nur darin, daß die schrittweise Abarbeitung, wenn eine Zeile erreicht wird, die einen Funktionsaufruf enthält, beim echten Einzelschritt (*Nächste Anweisung) in die Funktion hinein* fortgesetzt wird. Betrachten Sie noch einmal das vorstehende Programmlisting. Wenn Sie *Debug/Nächste Anweisung* (oder Taste [*F7*] oder den Schalter *Nächste Anweisung* der Mauspalette) zur schrittweisen Ausführung aller Anweisungen wählen würden, so würden Sie sehen, wie der Debugger nach dem Erreichen der Zeile, in der Rückgabewert der Funktion *InitAnwendung* ausgewertet wird, im nächsten Schritt den Ausführungsbalken in den Quelltext der Funktion springen läßt. Während des Debuggens können Sie einige Funktionen mit *Funktion überspringen* überspringen und in andere mit *Nächste Anweisung* hineingehen, ganz wie es erforderlich ist.

Schrittweise Abarbeitung bei Elementfunktionen von Klassen

Auch wenn Sie in Ihren Programmen Klassen verwenden, können Sie trotzdem den Debugger der IDE für *Funktion überspringen* und *Nächste Anweisung* verwenden. Der Debugger behandelt Elementfunktionen hinsichtlich schrittweiser Abarbeitung dabei genauso wie Funktionen in nicht-objektorientierten Programmen.

Schrittweise Abarbeitung bei externen Programmteilen

Wenn Sie externe Module zu Ihrem Programm hinzulinken, so können Sie auch diese mit *Funktion überspringen* und *Nächste Anweisung* debuggen, sofern die hinzugelinkten OBJ-Dateien Debug-Informationen enthalten.

Sie können externe Module jeder Programmiersprache debuggen, einschließlich C, C++, Pascal und Assembler. Solange diese Module alle Anforderungen an externes Linken erfüllen und vollständige symbolische Debug-Informationen nach Borland-Standard enthalten, kann der Debugger der IDE diese auch mit *Funktion überspringen* und *Nächste Anweisung* ablaufen lassen.

Große Programmteile mit *Funktion überspringen* überspringen

Manchmal möchten Sie nicht durch jede einzelne Zeile Ihres Quellprogramms gehen, nur um endlich an die Stelle zu gelangen, wo das Problem liegt. Der Debugger der IDE kann auch große Programmteile überspringen, also praktisch in einem Schritt ablaufen lassen, und Ihnen die Kontrolle dann an dem Punkt wiedergeben, ab dem Sie wieder in Einzelschritten weiterarbeiten wollen.

Lauf bis zu einem bestimmten Punkt

Sie können dem Debugger mitteilen, daß Sie Ihr Programm so lange normal (nicht schrittweise) ablaufen lassen wollen, bis dabei eine bestimmte Stelle in Ihrem Quellprogramm erreicht wird:

1. Setzen Sie den Cursor auf die Zeile, in der Sie wieder die Kontrolle zum Debuggen übernehmen wollen.
2. Wählen Sie *Gehe zur Cursorposition* in der Mauspalette (oder betätigen Sie [*F4*]).

Verwenden Sie den Befehl *Gehe zur Cursorposition* zum Beginnen Ihrer eigentlichen Debug-Sitzung, oder benutzen Sie ihn, nachdem Sie schon Einzelschritte haben ablaufen lassen.

Funktion suchen

Verwenden Sie den Befehl *Funktion suchen* im *Suchen*-Menü, um eine bestimmte Funktion schnell zu lokalisieren. Der Befehl fragt Sie nach dem Namen der Funktion und positioniert dann den Cursor auf die entsprechende Zeile der Quelldatei, wo die Definition dieser Funktion beginnt. Bevor Sie den Befehl *Funktion suchen* benutzen können, müssen Sie vorher bereits einmal Einzelschritte (per *Funktion überspringen* oder *Nächste Anweisung*) im Programm ausgeführt haben.

Rückkehr zur aktuellen Ausführungsposition

Während des Debuggens können Sie ungehindert alle Dateien in jedem Editierfenster betrachten, zu jeder Stelle in Ihrer Programmdatei gehen und sogar Dateien öffnen und schließen. Anschließend können Sie sehr schnell zur aktuellen Ausführungsposition zurückkehren.

Um zur aktuellen Ausführungsposition zu gelangen, wählen Sie *Debug/Ausführungsposition anzeigen*.

Der Debugger setzt den Cursor dann an die aktuelle Ausführungsposition. Falls Sie das Editierfenster, in dem diese Position liegt, inzwischen geschlossen haben, öffnet der Befehl *Debug/Ausführungsposition anzeigen* ein Editierfenster und zeigt den Quellprogrammteil an, in dem die Position zur Zeit liegt.

Rückwärts gehen

Während des Debuggens kann es nützlich sein, zu erfahren, wie man eigentlich an die augenblickliche Position gelangt ist. Das Fenster *Aufruf-Stack* zeigt Ihnen die Reihenfolge der Funktionsaufrufe, über die Sie an Ihre aktuelle Position gelangt sind. Wählen Sie *Anzeige/Aufruf-Stack*, um dieses Fenster zu öffnen.

Das Fenster *Aufruf-Stack* ist insbesondere dann nützlich, wenn Sie versehentlich per *Nächste Anweisung* in einen Programmteil gelaufen sind, den Sie eigentlich überspringen wollten. Mit folgender Methode können Sie zurückgehen und dann so wie ursprünglich gewollt mit *Funktion überspringen* und *Nächste Anweisung* arbeiten:

1. Doppelklicken Sie im Fenster *Aufruf-Stack* auf den Aufruf, der diejenige Funktion anspricht, in die Sie versehentlich mit *Nächste Anweisung* gelaufen sind. (Es ist dies im Fenster *Aufruf-Stack* der zweite Aufruf von oben.)
 Das Editierfenster wird dann zum aktiven Fenster, und der Cursor steht dort an der Stelle, wo der Aufruf ausgeführt wurde.
2. Bewegen Sie den Cursor im Editierfenster ganz hinter den Aufruf.
3. Wählen Sie *Gehe zur Cursorposition* in der Mauspalette des Editierfensters.

Das Fenster *Aufruf-Stack* ist außerdem nützlich, wenn Sie die beim Aufruf an eine Funktion übergebenen Parameter ansehen wollen.

Sie können den Quellprogrammteil, der einen bestimmten Aufruf enthält, ansehen oder verändern. Selektieren Sie dazu den Aufruf im Fenster *Aufruf-Stack*, und klicken Sie mit der rechten Maustaste, um die Mauspalette mit den Befehlen *Quelltext anzeigen* und *Quelltext bearbeiten* angezeigt zu bekommen.

Anhalten des Programms

Anstatt Ihr Programm mit *Funktion überspringen* oder *Nächste Anweisung* schrittweise ablaufen zu lassen, können Sie es auch einfach anhalten. Wählen Sie *Debug/Programm anhalten*, und der Programmablauf wird gestoppt. Nun können Sie die Werte von Variablen und Daten in diesem Programmzustand prüfen. Wenn Sie damit fertig sind, wählen Sie *Debug/Ausführen*, um die Ausführung Ihres Programms fortzusetzen.

Falls Ihr Programm aus irgendeinem Grunde die Kontrolle behält und nicht zum Debugger zurückkehrt (zum Beispiel, weil es in einer Endlos-

schleife läuft), können Sie die Tastenkombination [*Strg*][*Alt*][*S-Abf*] betätigen, um das Programm zu stoppen.

Neu starten

Gelegentlich möchten Sie während des Debuggens vielleicht vom Anfang an neu starten. Wählen Sie dazu *Debug/Programm beenden* oder betätigen Sie [*Strg*][*F2*]. Dadurch wird Ihr Programm abgebrochen, und der nächstfolgende Lauf- und Einzelschrittbefehl bezieht sich auf den Beginn der Funktion *main*.

Werte abfragen

Das Durchlaufen Ihres Programms in Einzelschritten kann Ihnen helfen, Fehler im Programmfluß aufzudecken, aber im Normalfall wollen Sie wissen, wie sich die Werte von Variablen bei den einzelnen Schritten verändern. Zum Beispiel ist es beim schrittweisen Durchlaufen einer **for**-Schleife hilfreich, den Wert der Zählvariablen zu kennen. Die IDE bietet Ihnen mehrere Hilfsmittel zur Überprüfung der Variablen Ihres Programms:

- Das Fenster *Überwachter Ausdruck* ermöglicht die Beobachtung der Werte von Variablen oder Ausdrücken.
- Das Dialogfenster *Ausdrucksauswertung* erlaubt Ihnen die Berechnung des Werts jedes im aktuellen Programm sinnvollen Ausdrucks; außerdem können Sie dort während des Debuggens Ihres Programms die Werte von Variablen verändern.
- Das Fenster *Variable untersuchen* läßt Sie die Werte von Datenelementen ansehen und ändern.

Was ist ein Ausdruck?

Die Funktionen *Beobachten*, *Berechnen* und *Ansehen* arbeiten auf der Ebene von *Ausdrücken*. Ein Ausdruck besteht aus Konstanten, Variablen und Datenstrukturen, verbunden durch Operatoren. Praktisch alles, was auf der rechten Seite einer Wertzuweisung zulässig ist, kann als Debug-Ausdruck verwendet werden.

Ausdrücke überwachen

Wenn Sie die Veränderung eines Werts einer Variablen oder eines Ausdrucks beobachten wollen, während Sie schrittweise durch Ihr Programm gehen, so sollten Sie dazu einen überwachten Ausdruck verwenden. Ein *überwachter Ausdruck* ist ein Ausdruck, den Sie im Fenster *Überwachter Ausdruck* eingeben; das Fenster zeigt danach ständig den aktuellen Wert des Ausdrucks an. Während Sie schrittweise durch Ihr Programm gehen, ändert sich der angezeigte Wert des überwachten Ausdrucks, sowie das Programm etwas zu seiner Änderung tut.

Wenn die Ausführungsposition des Programms aus dem Gültigkeitsbereichs eines überwachten Ausdrucks herausläuft, wird der Ausdruck undefiniert. Wenn sich das Programm später wieder in den Gültigkeitsbereich hineinbewegt, so wird der aktuelle Wert des Ausdrucks erneut angezeigt.

Um das Fenster *Überwachter Ausdruck* zu öffnen, wählen Sie *Anzeige/Überwachte Ausdrücke*. Solange Sie noch keinen überwachten Ausdruck eingegeben haben, ist das Fenster leer.

Bild 6.1
Das Fenster Überwachter Ausdruck



Einen überwachten Ausdruck hinzufügen

Um eine Variable im Fenster *Überwachter Ausdruck* hinzuzufügen, wählen Sie *Debug/Ausdruck hinzufügen*. Die IDE öffnet dann das Dialogfenster *Darstellung überwachter Ausdrücke* und läßt Sie einen überwachten Ausdruck eingeben.

Wie jedes andere Fenster der IDE können Sie auch das Fenster *Überwachter Ausdruck* verschieben, seine Größe ändern und es schließen. Ist das Fenster geschlossen, so lassen Sie es per *Anzeige/Überwachte Ausdrücke* wieder anzeigen. Wenn das Fenster *Überwachter Ausdruck* das aktive Fenster ist, können Sie über den Befehl *Ausdruck hinzufügen* in der Mauspalette einen überwachten Ausdruck hinzufügen.

Wenn das Editierfenster das aktive Fenster ist, können Sie den an der Cursorposition stehenden Ausdruck schnell in das Fenster *Überwachter Ausdruck* übertragen; wählen sie dazu *Überwachten Ausdruck festlegen* in der Mauspalette.

Formatieren von überwachten Ausdrücken

Wenn Sie im Fenster *Überwachter Ausdruck* auf einen überwachten Ausdruck doppelklicken, erscheint das Dialogfenster *Darstellung überwachter Ausdrücke*. Alternativ können Sie aber auch einen überwachten Ausdruck im Fenster *Überwachter Ausdruck* selektieren und dann in der Mauspalette dieses Fensters den Punkt *Set Properties* wählen.



Bild 6.2
Das Dialogfenster *Darstellung überwachter Ausdrücke*

Voreingestellt ist im Dialogfenster *Darstellung überwachter Ausdrücke* als Ausdruck das Wort an der Cursorposition im aktuellen Editierfenster. Eine Aufzeichnungsliste enthält die früher von Ihnen eingegebenen Watch-Ausdrücke.

Zum Formatieren der Ergebnisse von *Überwachte Ausdrücke*n selektieren Sie eine der Optionen im Fenster *Darstellung überwachter Ausdrücke*. Beispiel: Obwohl Werte vom ganzzahligen Typ normalerweise in dezimaler Form angezeigt werden, können Sie vorgeben, daß sie hexadezimal angezeigt werden, indem Sie den Schalter *Hexadezimal* aktivieren. Nachdem Sie *Dezimal* oder *Hexadezimal* gewählt haben, können Sie die Formatierung des angezeigten Ausdrucks mit den Optionen unter *Anzeigen als* noch genauer spezifizieren.

Wenn Sie eine Variable wie etwa ein Array im Fenster *Überwachter Ausdruck* beobachten, können Sie sich die Werte ihrer einzelnen Elemente anzeigen lassen. Für ein Feld *xarray* aus 5 Werten vom Typ **int** geben Sie im Dialogfenster *Darstellung überwachter Ausdrücke* im Feld *Wdh-Zähler* zum Beispiel die Zahl 5 ein, Sie bekommen dann alle 5 Werte des Array angezeigt. Ein Ausdruck, bei dem *Wdh-Zähler* benutzt wird, muß ein einzelnes Datenelement darstellen. Der Debugger betrachtet das Datenelement stets als erstes Element eines Array (falls das Element kein Zeiger ist) oder als Zeiger auf ein Array (falls es ein Zeiger ist).

Wenn Sie die Ansichtsoption *Gleitkomma* gewählt haben, können Sie auch vorgeben, mit wie vielen signifikanten Ziffern der Wert des überwachten Ausdrucks angezeigt werden soll. Geben Sie diese Ziffernanzahl im Feld *Signifikante Ziffern* ein.

Deaktivieren eines überwachten Ausdrucks

Wenn Sie den Wert eines überwachten Ausdrucks, den Sie ins Fenster *Überwachter Ausdruck* eingetragen haben, nicht mehr ständig sehen wollen, andererseits den Ausdruck aber nicht löschen möchten, weil Sie ihn vielleicht später nochmals benötigen, dann deaktivieren Sie einfach den überwachten Ausdruck.

Wenn die IDE viele überwachte Ausdrücke im Fenster *Überwachter Ausdruck* auswerten muß, kann die Einzelschrittverarbeitung dadurch verlangsamt werden. Dann ist es nützlich, einen überwachten Ausdruck deaktivieren und später bei Bedarf wieder aktivieren zu können. Um einen überwachten Ausdruck zu deaktivieren, klicken Sie auf das Markierungsfeld neben den überwachten Ausdruck. Um ihn wieder zu aktivieren, klicken Sie nochmals darauf.

Alternativ können Sie überwachte Ausdrücke auch mit Befehlen der Mauspalette deaktivieren und aktivieren. Um mehrere überwachte Ausdrücke zugleich zu deaktivieren oder zu aktivieren, klicken und ziehen Sie im Fenster *Überwachter Ausdruck* über die Gruppe der Ausdrücke, um sie zu selektieren, oder halten Sie die Taste [*Strg*] gedrückt, während Sie auf alle ü*berwac*hten Ausdrücke klicken, die Sie deaktivieren oder aktivieren wollen. Wählen sie anschließend den entsprechenden Befehl aus der Mauspalette.

Löschen eines überwachten Ausdrucks

Um einen überwachten Ausdruck zu löschen, selektieren Sie ihn im Fenster *Überwachter Ausdruck*, und wählen Sie dann *Ausdruck löschen* in der Mauspalette. Um alle überwachten Ausdrücke zu löschen, wählen Sie dort *Alle Ausdrücke löschen*.

Editieren eines überwachten Ausdrucks

Um die Attribute eines überwachten Ausdrucks zu ändern, lassen Sie sich das Dialogfenster *Darstellung überwachter Ausdrücke* für diesen Ausdruck anzeigen, und ändern Sie dann dort das gewünschte Attribut.

Berechnen und Verändern von Ausdrücken

Zusätzlich zu der Beobachtung von Variablen während des Ablaufs Ihres Programms können Sie jederzeit Ausdrücke berechnen und die Werte von Variablen zur Laufzeit ändern.

Auswerten von Ausdrücken

Um einen Ausdruck auszuwerten, wählen Sie *Debug/Auswerten/Ändern*; der Debugger zeigt dann die Dialogbox *Ausdrucksauswertung* an. In der Voreinstellung wird das Wort an der aktuellen Cursorposition im Editierfenster in der Dialogbox *Ausdrucksauswertung* farblich hervorgehoben. Sie können den Ausdruck bearbeiten, einen anderen eingeben oder aus der Aufzeichnungsliste einen früher schon berechneten auswählen.

Wenn Sie *Eval* aktivieren, erscheint der aktuelle Wert des Ausdrucks im Feld *Ergebnis*. Sie können jeden zulässigen C- oder C++ Ausdruck berechnen lassen, solange dieser nicht folgendes enthält:

- mit **#define** definierte Symbole oder Makros
- lokale oder statische Variablen, die nicht im Gültigkeitsbereich der gerade in Ausführung befindlichen Funktion liegen
- Funktionsaufrufe

Um Ausdrücke formatieren zu lassen, setzen Sie ein Komma, gefolgt von Formatspezifikatoren, dahinter. Um beispielsweise ein Ergebnis in hexadezimaler Form anzuzeigen, schreiben Sie **,H** hinter den Ausdruck. Tabelle 6.1 zeigt alle zulässigen Formatspezifikatoren und ihre Wirkung.

Tabelle 6.1: Formatspezifikatoren für Debug-Ausdrücke

Zeichen	Wirkt auf	Funktion
H oder **X**	Ganzzahlen	**Hexadezimal**. Zeigt ganzzahlige Werte, auch solche in Datenstrukturen, hexadezimal mit dem Präfix **0x** an.
C	Zeichen, Strings	**Zeichen**. Für die ASCII-Werte 0 bis 31 werden spezielle Zeichen angezeigt. Standardmäßig werden diese Zeichen durch die entsprechenden Escape-Sequenzen von C (**\n**, **\t** etc.) dargestellt.
D	Ganzzahlen	**Dezimal**. Zeigt ganzzahlige Werte, auch solche in Datenstrukturen, dezimal an.
F*n*	Gleitkommazahlen	**Gleitkomma**. Zeigt *n* signifikante Ziffern an (*n* = 2 bis 18, Standardwert 7)
*n***M**	Alle Typen	**Speicherabbild**. Zeigt *n* Bytes, beginnend mit der Adresse des betreffenden Ausdrucks. Wird *n* nicht angegeben, so wird standardmäßig die Größe des Variablentyps angenommen. Standardmäßig wird jedes Byte in Form zweier Hexadezimalziffern angezeigt. Um dies zu ändern, können die Spezifikatoren **C**, **D**, **H** und **S** zusammen mit **M** verwendet werden.
P	Zeiger	**Zeiger**. Zeigt Zeiger in der Form *Segment:Offset* mit zusätzlicher Information über die Adresse, auf die der Zeiger zeigt, an. Sie bekommen den Speicherbereich, in dem das Segment liegt und, falls vorhanden, den Namen der Variablen, die beim Offset liegt, angezeigt.
R	Strukturen, Varianten	**Struktur/Variante**. Zeigt die Elementnamen und die Werte an. Beispiel: `(`*X*`:1;`*Y*`:10;`*Z*`:5)`.
S	Zeichen, Strings	**String**. Zeigt die ASCII-Zeichen 0 bis 31 als Escape-Sequenzen (wie in C) an. Nur zum Ändern von Speicherabbildern (siehe oben unter *n***M**) verwenden.

Modifizieren von Variablen

Während des Debuggens können Sie den Wert einer Variablen mit Hilfe des Dialogfensters *Ausdrucksauswertung* verändern. Geben Sie dazu den Variablennamen bei *Ausdruck*, und den neuen Wert bei *Neuer Wert* ein. Wenn Sie wollen, daß der geänderte Wert in Ihrem Programm wirksam wird, wählen Sie *Modify*; falls Sie dies nicht tun, verwirft der Debugger den neuen Wert beim Schließen des Dialogfensters.

Halten Sie sich stets folgende Punkte vor Augen, wenn Sie Werte von Variablen ändern:

- Sie können die Werte von einfachen Variablen sowie von Elementen von Arrays oder Strukturen ändern, nicht aber die Arrays und Strukturen selbst.
- Der Ausdruck im Feld *Neuer Wert* muß ein Resultat ergeben, das zu der Variablen, der Sie es zuweisen wollen, so kompatibel ist, daß man es mit dem Zuweisungsoperator = zuweisen könnte. Eine gute Faustregel ist es, daß der neue Wert nicht zulässig ist, wenn er bei einer Zuweisung zu einem Compilier- oder Laufzeitfehler führen würde.
- Sie können Argumente ohne bestimmten Typ, die an eine Funktion übergeben werden, nicht direkt ändern; Sie können Sie jedoch mit dem Typumwandlungsoperator (**cast**) versehen und ihnen dann neue Werte geben.
- Benutzen Sie das Dialogfenster *Ausdrucksauswertung*, um Werte in Registern (einschl. Flag-Register) auszulesen oder zu ändern. Dazu können Sie Ausdrücke wie beispielsweise `_CS`, `_BX` oder `_FLAGS` eingeben. Die entsprechenden Werte sind Bitmuster.

Achtung!

- Das Verändern von Werten, insbesondere von Zeigerwerten und Array-Indizes, kann unerwünschte Wirkungen haben, denn Sie überschreiben damit vielleicht anderen Variablen und Datenstrukturen. Seien Sie damit sehr vorsichtig.

Datenelemente ansehen

Die Werte von Datenelementen können Sie in einem *Inspector*-Fenster ansehen und ändern. Dazu tun Sie folgendes:

1. Wählen Sie *Debug/Ansehen/Ändern*, um das Fenster *Data Inspektor* anzuzeigen.
2. Geben Sie den Ausdruck ein, den Sie anzeigen lassen wollen.
3. Wählen Sie *OK*, um ein *Inspector*-Fenster anzuzeigen.

Wenn sich die Ausführungsposition des Programms im Gültigkeitsbereich des Ausdrucks, den Sie inspizieren, befindet, erscheint der Wert des Ausdrucks im Fenster *Variable untersuchen*. Befindet sich die Position außerhalb des Bereichs, so ist der Wert undefiniert.

Sie können ein *Inspector*-Fenster auch direkt aus dem Editierfenster heraus anzeigen lassen:

4. Setzen Sie den Cursor auf das Datenelement, daß Sie ansehen wollen.
5. Wählen Sie *Objekt untersuchen* in der Mauspalette, oder betätigen Sie [*Return*].

Wenn Sie diese Methode wählen, wird der Wert des Datenelements stets innerhalb des Gültigkeitsbereichs der Zeile, in der es erscheint, berechnet.

Bei beiden Methoden richtet sich die Form, in der die Daten in einem *Inspector*-Fenster erscheinen, nach dem Typ der inspizierten Daten. Wenn Sie beispielsweise ein Array überprüfen, werden Sie für jedes Element des Array eine Zeile mit dem Index des Elements sehen; dahinter folgt der Wert des Elements in seinem Anzeigeformat, und dahinter wiederum der Wert in hexadezimaler Form.

Wenn Sie einmal in einem *Inspector*-Fenster sind, können Sie die Sicht eingrenzen und bestimmte Elemente genauer inspizieren. Um Einzelheiten eines Elements zu sehen, tun Sie folgendes:

6. Selektieren Sie das Element, daß Sie genauer ansehen wollen.
7. Betätigen Sie [*Return*].

Sie können auch die Werte von *Inspector*-Elementen verändern. Um den Wert eines einzelnen Elements zu verändern, tun Sie folgendes:

8. Selektieren Sie das Element.
9. Wählen Sie *Ändern* in der Mauspalette.
10. Geben Sie einen neuen Wert ein, und bestätigen Sie dann mit *OK*.

Wenn Sie eine Datenstruktur inspizieren, kann die Anzahl der anzuzeigenden Elemente so groß sein, daß Sie im *Inspector*-Fenster einen Bildlauf durchführen müssen, um die Sie interessierenden Daten zu sehen. Um mehr Übersicht zu haben, können Sie die Anzeige auf einen bestimmten Bereich der Datenelemente beschränken.

Um eine Gruppe von Elementen anzuzeigen, tun Sie folgendes:

11. Klicken Sie in das *Inspector*-Fenster.
12. Wählen Sie *Bereich* in der Mauspalette.
13. Geben Sie bei *Start-Index* den Index des Datenelements ein, welches das erste im Fenster sichtbare sein soll.
14. Geben Sie bei *CAnzahl* die Anzahl der Datenelemente ein, die Sie im *Inspector*-Fenster sehen wollen.

Registerwerte inspizieren

Während des Debuggens können Sie die Werte der Daten-, Pointer-, Index-, Segment- und Programmzähler-Registern sowie die Inhalte der Statusflags anzeigen lassen. Wählen Sie *Anzeige/Register*, um das Fenster *Register* zu sehen.

Sie können sich die Registerinhalte auch mit Hilfe des Dialogfensters *Ausdrucksauswertung* ansehen.

Tabelle 6.2: Im Fenster *Register* angezeigte CPU-Flags

Angezeigter Buchstabe	Flag
c	Carry
z	Zero
s	Sign
o	Overflow
p	Parity
a	Auxiliary carry
i	Interrupt enable
d	Direction

Die Mauspalette des Fensters *Register* erlaubt es Ihnen, das Anzeigeformat der Registerwerte zu wählen; selektieren Sie *Hexadezimal* oder *Dezimal*. Außerdem können Sie die Registeranzeige zwischen 16 Bit (Wort) und 32 Bit (Doppelwort) umschalten.

Haltepunkte verwenden

Ein *Haltepunkt* ist eine bestimmte Stelle in Ihrem Quellprogramm, an der (von Ihnen gewünscht) das Programm anhalten und die Kontrolle an den Debugger übergehen soll. Die Wirkung eines Haltepunkts ähnelt der des Befehls *Gehe zur Cursorposition*, denn das Programm läuft mit voller Geschwindigkeit, bis es den betreffenden Punkt erreicht. Im Gegensatz zum Befehl *Gehe zur Cursorposition* sind jedoch mehrere Haltepunkte möglich, und Sie können bestimmen, daß der Stop an einem Haltepunkt nur unter gewissen Bedingungen erfolgt.

Haltepunkte setzen

Um einen Haltepunkt in Ihr Programm zu setzen, müssen Sie zuerst den Cursor auf die Zeile stellen, wo der Stop erfolgen soll. In dieser Zeile muß etwas »Ausführbares« stehen; Kommentare, Leerzeilen oder Deklarationen werden vom Debugger als unzulässig reklamiert. Um einen Haltepunkt für die Zeile zu aktivieren, wählen Sie *Haltepunkt umschalten* ([*F5*]) im *Debug*-Menü oder in der Mauspalette des Editierfensters; die Zeile erscheint dann farblich hervorgehoben.

Wenn Sie nun Ihr Programm in der IDE ablaufen lassen, so wird es anhalten, sowie es diese Zeile erreicht, aber *bevor* es diese ausführt. Im Editier-

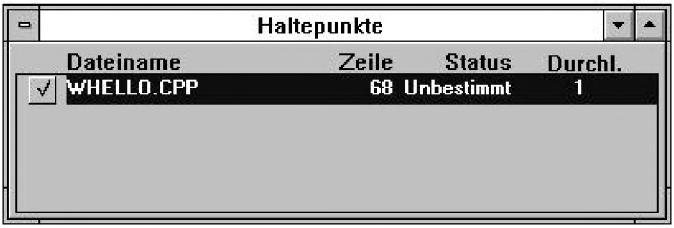
fenster erscheint die Zeile, die den Haltepunkt enthält, durch den Ausführungszeiger markiert. In dieser Situation können Sie nun jede beliebige Debug-Aktion durchführen, zum Beispiel Einzelschrittbetrieb (*Funktion überspringen* und *Nächste Anweisung*), Aktivieren von *Überwachte Ausdrücke*n, Arbeiten mit einem *Inspector*-Fenster sowie Durchführen von Berechnungen.

Um einen Haltepunkt zu löschen, stellen Sie den Cursor auf die Zeile, auf die er gesetzt ist; wählen Sie dann *Haltepunkt umschalten* ([*F5*]) im *Debug*-Menü oder in der Mauspalette des Editierfensters.

Arbeiten mit Haltepunkten

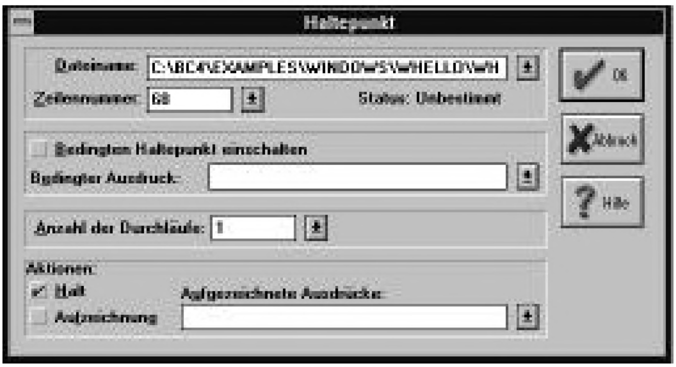
Die IDE merkt sich während einer Debug-Sitzung alle Ihre Haltepunkte und ordnet sie Ihrem aktuellen Projekt zu. Sie können alle Ihre Haltepunkte von einem einzigen Fenster aus kontrollieren und müssen nicht in allen Quelldateien nach ihnen suchen. Wählen Sie *Anzeige/Haltepunkt*, um das Fenster *Haltepunkte* zu öffnen.

Bild 6.3: Das Fenster Haltepunkte



Im Fenster *Haltepunkte* können Sie bestimmte Attribute für jeden Haltepunkt aktivieren und ansehen. Doppelklicken Sie dazu auf den betreffenden Haltepunkt im Fenster *Haltepunkte*, oder selektieren Sie ihn und wählen dann *Haltepunkt bearbeiten* in der Mauspalette. Es erscheint dann das Dialogfenster *Haltepunkt*.

Bild 6.4: Das Dialogfenster Haltepunkt



Haltepunkte löschen

Um einen Haltepunkt zu löschen, selektieren Sie ihn im Fenster *Haltepunkte* und wählen dann *Haltepunkt löschen* in der Mauspalette. Um alle Haltepunkte zu löschen, wählen Sie *Alles Löschen* in der Mauspalette.

Alternativ können Sie Haltepunkte auch löschen, indem Sie einen der Befehle *Haltepunkt löschen* oder *Alles löschen* in der Mauspalette des Dialogfensters *Haltepunkt* verwenden.

Aktivieren und Deaktivieren von Haltepunkten

Sie können Haltepunkte auch zeitweise deaktivieren. Der einfachste Weg dazu besteht im Deaktivieren des Markierungsfelds des Haltepunkts, den Sie deaktivieren wollen, im Fenster *Haltepunkte*. Wenn Sie diesen Haltepunkt später wieder deaktivieren wollen, klicken Sie wieder auf sein Markierungsfeld.

Alternativ können Sie Haltepunkte auch mit Befehlen der Mauspalette aktivieren und deaktivieren. Um mehrere Haltepunkte zugleich zu deaktivieren oder zu aktivieren, klicken und ziehen Sie im Fenster *Haltepunkte* über die Gruppe der Haltepunkte, um sie zu selektieren, oder halten Sie die Taste [*Strg*] gedrückt, während Sie auf alle Haltepunkte klicken, die Sie deaktivieren oder aktivieren wollen. Wählen sie anschließend den entsprechenden Befehl aus der Mauspalette.

Ansehen und Editieren des Programms an einer Haltepunkt-Position

Sie können einen Haltepunkt in Ihrem Quellprogramm selbst dann sehr schnell finden, wenn er sich nicht in Ihrem aktuellen Editierfenster befindet. Selektieren Sie dazu den Haltepunkt im Fenster *Haltepunkte*, und wählen Sie dann *Quelltext anzeigen* in der Mauspalette. Daraufhin erscheint der Haltepunkt im aktuellen Editierfenster, wobei der Cursor auf ihn gesetzt ist. Das Fenster *Haltepunkte* bleibt weiterhin als das aktive Fenster geöffnet.

Wenn Sie die Zeile Ihres Quellprogramms, auf der ein Haltepunkt steht, nicht nur ansehen, sondern *editieren* möchten, wählen Sie in der Mauspalette des Fensters *Haltepunkt* den Befehl *Quelltext bearbeiten*. Der Haltepunkt erscheint dann im aktuellen Editierfenster, wobei der Cursor auf ihn gesetzt ist, und Sie können dort sofort editieren.

Ungültige Haltepunkte löschen

Haltepunkte müssen auf ausführbare Anweisungen gesetzt werden, an anderen Stellen sind sie ungültig. Zum Beispiel ist ein Haltepunkt, der auf eine leere Zeile oder eine Deklaration gesetzt ist, ungültig. Wenn Sie versehentlich einen ungültigen Haltepunkt gesetzt haben und Ihr Programm dann (weiter)laufen lassen, prüft der Debugger alle Haltepunkte und meldet einen ungültigen durch Öffnen des Dialogfensters *Ungültiger Haltepunkt*. Schließen Sie dieses Fenster, öffnen Sie das Fenster *Haltepunkte*, suchen Sie dort den ungültigen Haltepunkt, und löschen Sie ihn. Wenn Sie den Haltepunkt an eine zulässige Stelle setzen wollen, können Sie dies jetzt tun. Lassen Sie anschließend Ihr Anwendungsprogramm weiterlaufen.

Eine andere Methode ist es, das Fenster *Ungültiger Haltepunkt* zu ignorieren, es zur Seite zu schieben und den Befehl *Ausführen* zu wählen, um das

Anwendungsprogramm fortzusetzen. Die IDE löscht dann den ungültigen Haltepunkt und läßt Ihr Programm weiterlaufen.

Ansehen und Ändern von Haltepunkt-Attributen

Um die Attribute für einen Haltepunkt anzusehen oder zu ändern, doppelklicken Sie auf den betreffenden Haltepunkt (oder selektieren Sie ihn) im Fenster *Haltepunkte* und wählen dann *Haltepunkt bearbeiten* in der Mauspalette. Daraufhin erscheint das Dialogfenster *Haltepunkt*.

Außer zum Prüfen der Attribute eines Haltepunkts können Sie das Dialogfenster *Haltepunkte*noch für andere Dinge wie Setzen eines neuen Haltepunkts, Modifizieren eines vorhandenen Haltepunkts sowie Umwandeln eines Haltepunkts in einen bedingten Haltepunkt benutzen.

Die folgenden Abschnitte erläutern, wie dies zu tun ist.

Setzen eines neuen Haltepunkts

Obwohl es am einfachsten ist, einen Haltepunkt mit Hilfe des Befehls *Haltepunkt umschalten* in der Mauspalette des Editierfensters zu setzen, möchten Sie ihn manchmal vielleicht mit Hilfe des Dialogfensters *Haltepunkte* setzen. Letzteres ist besonders dann zu empfehlen, wenn Sie einen bedingten Haltepunkt setzen wollen:

1. Stellen Sie den Cursor an die Stelle, wo Sie den Haltepunkt setzen wollen.
2. Wählen Sie *Haltepunkt hinzufügen* entweder im *Debug*-Menü oder in der Mauspalette des Fensters *Haltepunkte*.

 Es erscheint dann das Dialogfenster *Haltepunkte*.

Es ist auch möglich, einen Haltepunkt in einer Datei zu setzen, für die zur Zeit kein Editierfenster geöffnet ist:

3. Geben Sie im Dialogfenster *Haltepunkte* den Namen der Datei ein, in der Sie den Haltepunkt setzen wollen.
4. Geben Sie im Feld *Linie* die Nummer der Zeile in Ihrem Quellprogramm ein, wo Sie den Haltepunkt setzen wollen.

Modifizieren eines vorhandenen Haltepunkts

Um die Attribute eines vorhandenen Haltepunkts zu verändern, gehen Sie in das Feld *Haltepunkte*, und geben Sie dort neue Werte für diejenigen Attribute ein, die Sie ändern wollen.

Setzen von Haltepunkten, während das Programm läuft

Um einen Haltepunkt zu setzen, während Ihr Programm läuft, schalten Sie einfach (wie auf irgendeine andere Windows-Anwendung) auf den Debugger um. Wenn Sie Ihr Programm dann weiterlaufen lassen, wird der neue Haltepunkt gesetzt, und das Programm stoppt, wenn er erreicht wird.

Bedingte Haltepunkte setzen

Die Haltepunkte, die Sie mit dem Befehl *Haltepunkt umschalten* in der Mauspalette des Editierfensters setzen, sind unbedingte Haltepunkte: jedesmal, wenn diese Zeile erreicht wird, stoppt der Debugger das Programm. Wenn Sie die Attribute eines neuen oder schon vorhandenen Haltepunkts editieren, haben Sie jedoch zusätzliche Optionen im Dialogfenster *Haltepunkte*, mit denen Sie *bedingte* Haltepunkte erzeugen können. Es gibt zwei Arten von Bedingungen für Haltepunkte: *Bedingter Ausdruck* und *Anzahl der Durchläufe*.

Sie können einen booleschen Ausdruck als Bedingung für einen Haltepunkt eingeben. Damit können Sie beispielsweise testen, ob der Wert einer Variablen in einem bestimmten Bereich liegt oder ob ein bestimmter Merker gesetzt ist. Wenn die vorgegebene Bedingung erfüllt ist, unterbricht der Debugger den Programmablauf und übergibt Ihnen die Kontrolle.

Wenn Sie eine Zahl von Durchgängen für einen Haltepunkt angeben, weisen Sie den Debugger damit an, das Programm nicht jedesmal zu unterbrechen, wenn die vorgegebene Bedingung erfüllt ist, sondern erst dann, wenn sie das *n*-te Mal erfüllt ist. Wenn also für die Anzahl von Durchgängen 3 vorgegeben ist, unterbricht der Debugger das Programm erst, wenn er beim dritten Mal die Bedingung als erfüllt erkennt. Wenn Sie keine Bedingung vorgeben, unterbricht der Debugger das Programm jedesmal, wenn die Haltepunkt-Zeile erreicht wird.

Sie können auf Wunsch den Wert eines bestimmten Ausdrucks jedesmal dann in das Fenster Ereignisse schreiben lassen, wenn ein Haltepunkt erreicht wird:

1. Wählen Sie *Aufzeichnung* im Dialogfenster *Haltepunkt*.
2. Geben Sie im *Aufgezeichnete Ausdrücke* den Ausdruck ein, dessen Wert Sie berechnet haben möchten.

Beispiel: Wenn Sie den Namen einer Variablen im Feld *Aufgezeichnete Ausdrücke* des Dialogfensters *Haltepunkt* eingeben, so schreibt der Debugger jedesmal, wenn er den von Ihnen gesetzten Haltepunkt erreicht, den Wert dieser Variablen in das Fenster *Ereignisse*.

Anpassung der Farben für Haltepunkte und Ausführungszeiger

Sie können eigene Farben vorgeben, in denen aktivierte, deaktivierte und ungültige Haltepunkt-Zeilen angezeigt werden:

1. Wählen Sie *Optionen/IDE* und dann *Syntax-Hervorhebung/Konfigurieren*.
2. Wählen Sie in der Liste *Element* diejenige Haltepunkt-Art, die Sie ändern wollen, und verändern Sie dann Vordergrund- und Hintergrundfarbe nach Ihren Wünschen.

Sie können auch die Farbe des Ausführungszeigers bestimmen; wählen Sie dazu das Element *Ausführungsposition* und danach die gewünschten Farben.

Abfangen des Fehlers Allgemeine Schutzverletzung

Wenn ein GPE-Fehler (General Protection Exception = Allgemeiner Schutzverletzungs-Fehler) auftritt, während Ihr Programm unter Kontrolle der IDE abläuft, so wird es gestoppt, und es erscheint das Dialogfenster *Allgemeine Schutzverletzung*. Wenn Sie das Fenster durch Wählen von *OK* schließen, zeigt der Debugger anschließend die Anweisungen an der Stelle, wo die Schutzverletzung stattfand. Der Ausführungsbalken markiert die »schuldige« Anweisung.

In dieser Situation müssen Sie *Debug/Programm beenden* wählen, um den Absturz Ihres Programms zu verhindern. Korrigieren Sie danach den Fehler, der das Problem verursacht hat, und starten Sie Ihr Programm erst dann neu.

Der Debugger kann zwar die meisten GPE-Fehler abfangen, je nach konkreter Ursache gelingt ihm dies manchmal aber nicht.

Benutzen des Fensters Ereignisse

Um das Fenster Ereignisse anzuzeigen, wählen Sie *Anzeige/Ereignisse*.

Weiter oben wurde beschrieben, wie die Werte von Ausdrücken, die Sie spezifiziert haben, im Fenster *Ereignisse* aufgezeichnet werden können, sobald der Debugger einen Haltepunkt erreicht.

Sie können bestimmen, daß auch Windows-Botschaften und Programmausgaben im Fenster *Ereignisse* aufgezeichnet werden. Um zu bestimmen, welche Ereignisse aufgezeichnet werden sollen, wählen Sie den Befehl *Optionen einstellen* in der Mauspalette des Fensters *Ereignisse*. Selektieren Sie den Punkt *Debug* und dann die gewünschten Optionen unter *Event Capture*.

Die folgende Tabelle beschreibt alle Befehle der Mauspalette des Fensters *Ereignisse*:

Tabelle 6.3: Befehle der Mauspalette des Fensters *Ereignisse*

Befehl	Beschreibung
Ereignisse speichern	Zeigt ein Dialogfenster zur Eingabe eines Dateinamens an. Alle im Fenster *Ereignisse* erscheinenden Ereignisse werden in diese Datei geschrieben, so daß Sie sie später untersuchen können.

Kommentar hinzufügen	Ermöglicht Ihnen die Eingabe einer Textzeile, die dann als Kommentar in das Fenster *Ereignisse* eingefügt wird.
Ereignisse löschen	Löscht den Inhalt des Fensters *Ereignisse*.
Optionen einstellen	Zeigt das Dialogfenster *IDE-Einstellungen* an, wo Sie Debug-Optionen und *Event Capture*-Optionen wählen können.

Debuggen von dynamischen Linkbibliotheken (DLLs)

Wenn Sie per *Funktion überspringen* oder *Nächste Anweisung* in eine von Ihrer Anwendung benutzte DLL laufen, so lädt der Debugger automatisch die Symboltabelle der DLL, also die Liste, die er benutzt, um alle in der DLL verwendeten Variablen, Konstanten, Typen und Funktionsnamen zu überwachen.

Zu jedem Zeitpunkt kann immer nur eine einzige Symboltabelle in den Speicher geladen sein; deshalb wird die Symboltabelle Ihrer EXE-Datei ausgelagert, wenn die Tabelle der DLL geladen wird. Aus diesem Grund können Sie dann keine Variablenwerte überwachen und keine Daten ansehen u.ä., die zum Quelltext Ihrer EXE-Datei gehören. Natürlich können Sie diese Tätigkeiten in dieser Situation bezogen auf das DLL-Quellprogramm durchführen.

Um festzustellen, mit welcher Symboltabelle der Debugger gerade arbeitet, wählen Sie *Debug/Symboltabelle laden*; sie bekommen dann die aktuelle Tabelle sowie auch alle anderen verfügbaren gezeigt. Beispiel: Wenn Ihre EXE-Datei MEINEANW.EXE heißt und die DLL MEINEDLL.DLL benutzt, so wird die Symboltabelle von MEINEDLL.DLL als aktuelle und die Tabelle von MEINEANW.EXE als verfügbare angezeigt.

Sie können auch zwischen den Symboltabellen wechseln. Wenn Sie beispielsweise gerade MEINEDLL.DLL in Einzelschritten durchlaufen und dabei den Wert einer Variablen, die in MEINEANW.EXE liegt, überprüfen wollen, so können Sie dies nach Laden der Symboltabelle von MEINEANW.EXE tun:

1. Wählen Sie *Debug/Symboltabelle laden*.
2. Selektieren Sie die gewünschte Symboltabelle in der Symboltabellenliste *Verfügbar*.

Nachdem Sie den Wert der Variablen überprüft haben, müssen Sie wieder die ursprüngliche Symboltabelle zurückladen, wenn Sie den schrittweisen Ablauf wieder aufnehmen wollen.

Soft- und Hard-Modus des Debuggers

Windows verarbeitet und erzeugt ständig diverse Botschaften. Jedes laufende Windows-Programm, wie etwa das Borland C++ für Windows, muß dies ebenfalls tun. Wenn Sie Ihre eigenen Windows-Programme ablaufen lassen, senden auch diese Botschaften ab und verarbeiten empfangene.

Wenn Sie in Ihrem Programm einen Haltepunkt setzen, wollen Sie aber ja gerade, daß Ihr Programm stoppt, wenn es den Haltepunkt erreicht. In der Windows-Umgebung sendet Windows aber schon dann Botschaften an Ihr laufendes Programm, wenn lediglich Aktionen wie etwa das Verändern der Fenstergröße, das Öffnen eines Dialogfensters, Bewegen des Mauszeigers oder das Öffnen eines Menüs stattfinden. Falls der Debugger diese Botschaften nicht für Sie verarbeitet, könnten sie bewirken, daß Ihr Programm weiterläuft oder daß das System blockiert.

Glücklicherweise nimmt Ihnen der IDE-Debugger die Behandlung der meisten solcher Situationen ab, und Sie bzw. Ihr Programm können während des Debuggens alle Funktionen der Windows-Umgebung nutzen. Diese Betriebsart des Debuggers, in der er gleichzeitig mit anderen Windows-Programmen läuft, heißt *Soft-Modus*.

Wegen gewisser Beschränkungen von Windows kann es manchmal jedoch erforderlich sein, alle anderen Windows-Programme außer dem Debugger selbst anzuhalten. Diese Betriebsart heißt *Hard-Modus*. Wenn sich der Debugger im Hard-Modus befindet, können Sie die Funktionalität der Windows-Umgebung nicht nutzen. Sie können dann beispielsweise nicht auf eine andere Task umschalten. Der Debugger verhält sich dann so, als wäre er das einzige Programm im Speicher.

Sie werden wahrscheinlich Ihre Programme im Hard-Modus debuggen wollen, wenn sie Botschaften (etwa DDE-Botschaften) an andere Tasks senden oder wenn Sie andere Debug-Aktionen auf sehr niedriger Systemebene durchführen müssen. Um den Hard-Modus zu aktivieren, tun Sie folgendes:

1. Wählen Sie *Optionen/IDE*, um das Dialogfenster *IDE-Einstellungen* zu öffnen.
2. Wählen Sie *Debugger*.
3. Wählen Sie die Option *Harter Modus*.

Wenn Sie die Option *Automatisch* aktivieren, wechselt der Debugger selbst zwischen Hard- und Soft-Mode, je nachdem, was in der Windows-Umgebung abläuft. Dies ist die Option, die sie wahrscheinlich am häufigsten benutzen werden. Um die Option *Automatisch* zu aktivieren, tun Sie folgendes:

4. Wählen Sie *Optionen/IDE*.
5. Wählen Sie *Debugger*.
6. Wählen Sie die Option *Smart*.

K A P I T E L 7

WinSight

WinSight ist ein Debugger-Werkzeug, das Informationen über Fenster, Fensterklassen und Botschaften bereitstellt. Sie können es zur Untersuchung einer Windows-Anwendung verwenden (Ihrer Anwendung oder einer anderen), um zu ermitteln, wie Fenster und Fensterklassen erzeugt und benutzt werden und welche Botschaften die Fenster erhalten.

Sie können *WinSight* so konfigurieren, daß Botschaften

- von Fenstern
- von Fensterklassen
- nur von bestimmten Typen
- oder alle

verfolgt werden können. *WinSight* ist ein passiver Beobachter: Es fängt Informationen über Botschaften ab und zeigt sie an, verhindert aber deren Weitergabe an andere Anwendungen nicht.

WinSight starten



Mit einem Doppelklick auf sein Symbol starten Sie *WinSight*. Es erscheint dann das *WinSight*-Fenster in seiner Standardkonfiguration. Das Fenster zeigt den Ausschnitt *Fensterhierarchie* mit allen momentan auf dem Desktop geöffneten Fenstern. *WinSight* speichert Ihre Konfiguration; wenn Sie also zum Beispiel alle drei Ausschnitte öffnen und dann *WinSight* verlassen, so erscheinen beim nächsten Start wieder alle drei.

WinSight kennt drei Ausschnitte: *Klassenliste*, *Fensterhierarchie* und *Botschaftsverfolgung*. Um diese Ausschnitte von links nach rechts oder von oben nach unten auf dem Bildschirm anzuordnen, wählen Sie *Anzeige/Horizontal anordnen* oder *Anzeige/Vertikal anordnen*.

Sie können festlegen, welche Botschaften *WinSight* verfolgen soll, und Sie können außerdem bestimmen, wann diese Verfolgung beginnen und enden soll (siehe hierzu die folgenden Abschnitte).

Bildschirmaktualisierung freigeben und anhalten

Zum Starten der Verfolgung wählen Sie den Menübefehl *Start!* (der Menüpunkt *Start!* wechselt dann auf *Stop!*). Im Normalfall werden alle drei Ausschnitte laufend aktualisiert, wenn Klassen registriert, Fenster erzeugt oder freigegeben und Botschaften empfangen werden. Sie können jedoch durch *Botschaften/Aufzeichnung aus* die Botschaftsverfolgung einzeln ausschalten (*Klassenliste* und *Fensterhierarchie* werden weiterhin aktualisiert).

Benutzen Sie *Stop!* und *Start!* für

- die Untersuchung einer bestimmten Situation
- für die Reduzierung der durch die ständige Aktualisierung von *WinSight* verbrauchten Rechenzeit.

Botschaftsverfolgung abschalten

Durch Auswahl von *Botschaften/Aufzeichnung aus* wird die Botschaftsverfolgung ausgeschaltet. Der Ausschnitt *Botschaften* bleibt sichtbar und die Verfolgung wird wieder aufgenommen (vorausgesetzt, die Verfolgung ist aktiviert), wenn Sie eine Menüoption zur Botschaftsverfolgung auswählen: *Ausgewählte Klassen*, *Ausgewählte Fenster* oder *Alle Fenster*.

Die folgenden Abschnitte beschreiben, wie Sie die drei Ausschnitte benutzen können, um die bei der Fehlersuche in einem Anwendungsprogramm nötigen Informationen zu erhalten. Zum Verlassen von *WinSight* wählen Sie den Befehl *Überwachung/Beenden*.

Einen Ausschnitt auswählen

WinSight bietet drei Ausschnitte an, die in seinem Hauptfenster erscheinen können: *Fensterhierarchie*, *Klassenliste* und *Botschaftsverfolgung*. Wählen Sie selbst, ob nur einer oder mehrere Ausschnitte gleichzeitig angezeigt werden sollen.

WinSight ordnet die Ausschnitte automatisch innerhalb des Hauptfensters an.

- Der Ausschnitt *Klassenliste* zeigt alle aktuell registrierten Fensterklassen an.
- Der Ausschnitt *Fensterhierarchie* zeigt die Hierarchie der Fenster auf dem Desktop an. Das ist die voreingestellte Darstellung, wenn Sie *WinSight* starten.

- Der Ausschnitt zur Botschaftsverfolgung zeigt Informationen über Botschaften an, die von ausgewählten Fenstern oder Fensterklassen empfangen werden.

Weitere Einzelheiten über ein Element im Ausschnitt *Fensterhierarchie* oder *Klassenliste* erfahren Sie durch

- Aktivieren von *Verfolgung/Detailinformationen*, nachdem Sie ein Fenster oder eine Klasse gewählt haben
- Doppelklick auf das Fenster oder die Klasse.

Das Detailfenster zeigt den Klassennamen, das Programmodul und andere Informationen über die Klasse oder das Fenster.

Klassenliste

Eine Klasse ist der Name, mit dem die Fensterklasse mit Windows registriert wurde.

Manchmal wollen Sie nicht nur die Botschaften für ein spezielles Fenster, sondern für ganze Fensterklassen verfolgen. *WinSight* bietet Ihnen dies mit Hilfe des Ausschnitts *Klassenliste*.

Den Ausschnitt *Klassenliste* benutzen

Der Ausschnitt *Klassenliste* zeigt alle momentan registrierten Fensterklassen. Rufen Sie Details über eine Klasse durch Doppelklick oder durch Betätigen der [*Eingabetaste*], wenn die Klasse ausgewählt wurde, ab.

Die Rhomben neben den Klassen werden jedesmal kurzzeitig schwarz, wenn das Fenster irgendeine Botschaft erhält. Dadurch erhalten Sie einen Überblick darüber, welche Fenster momentan Botschaften erhalten. Wenn ein unsichtbares untergeordnetes Fenster eine Botschaft erhält, ändert der Rhombus des übergeordneten Fensters seine Farbe.

Format

```
Klasse {Modul} Funktion Stile
```

Klasse ist der Name der Klasse. Einige vordefinierte Windows-Klassen tragen numerische Namen. So heißt die Pop-up-Menüklasse beispielsweise *32768*. Diese vordefinierten Klassen werden sowohl mit ihrer Nummer als auch mit ihrem Namen angezeigt, z.B. `#32768:PopupMenu`. Der eigentliche Klassenname ist allein die Nummer (angegeben im MAKEINTRESOURCE-Format, das auch für Ressourcen-IDs verwendet wird).

Modul ist der Name des ausführbaren Moduls (.EXE oder .DLL), welches die Klasse registriert hat.

Funktion ist die Adresse der Klassenfensterfunktion.

Stile ist eine Liste der *cs_*-Stile für die Klasse. Die Namen entsprechen den *cs*-Definitionen in *WinTypes*, mit der Ausnahme, daß *cs_*weggelassen wird und der Name in gemischter Groß-/Kleinschreibung dargestellt ist.

Klassen untersuchen

Um Botschaften für eine oder mehrere Klassen zu verfolgen, wählen Sie die Klassen in *Klassenliste* aus ([*Umschalt*] [*Klick*] oder [*Strg*] [*Klick*]), und wählen Sie dann *Botschaften/Ausgewählte Klassen*. Falls der Ausschnitt *Botschaften* unsichtbar ist, wenn Sie *Botschaften/Ausgewählte Klassen* wählen, wird er daraufhin sichtbar.

Beachten Sie, daß das Verfolgen von Botschaften an eine Klasse Sie in die Lage versetzt, alle Botschaften an Fenster dieser Klasse anzusehen, einschließlich Initialisierungsbotschaften, die anderweitig nicht zugänglich sind.

Um andere als die zur Zeit verfolgten Klassen zu verfolgen, ändern Sie die Auswahl in der *Klassenliste*. Wählen Sie *Botschaften/Verfolgung aus*, um die Verfolgung sämtlicher Botschaften im Ausschnitt *Botschaftsverfolgung* zu deaktivieren.

Fensterhierarchie

Der Ausschnitt *Fensterhierarchie* zeigt die Hierarchie aller auf dem Desktop vorhandenen Fenster. Mit Hilfe dieser Ansicht können Sie:

- feststellen, welche Fenster auf dem Desktop vorhanden sind,
- den Status von Fenstern überprüfen, einschließlich versteckter Fenster,
- feststellen, welche Fenster Botschaften empfangen,
- Fenster auswählen, deren Botschaften verfolgt werden sollen.

Die Linien auf der linken Seite des Ausschnitts *Fensterhierarchie* zeigen die Baumstruktur. Sie verbinden jedes Fenster mit seinem übergeordneten und seinen gleich- und untergeordneten Fenstern. Der Rhombus neben jedem Fenster zeigt an, ob das Fenster untergeordnete Fenster hat. Wenn ein Fenster eine Botschaft empfängt, wird der ihm (oder seinem übergeordneten Fenster, wenn der Baum zugeklappt ist) nächststehende Rhombus schwarz.



Das Fenster hat keine untergeordneten Fenster.



Das Fenster hat untergeordnete Fenster, sie werden aber momentan nicht angezeigt. Um die nächste untergeordnete Ebene anzuzeigen, klicken Sie auf den dem Fenster nächstgelegenen Rhombus.



Das Fenster hat untergeordnete Fenster, die angezeigt werden (mindestens eine Fensterebene niedriger ist sichtbar, weitere sind möglicherweise zugeklappt). Um *alle* untergeordneten Fenster eines Fensters unsichtbar zu machen, klicken Sie (einfach oder doppelt) auf den dem Fenster nächstgelegenen Rhombus.

Format `Handle {Klasse} Modul Position "Titel"`

Handle ist das Handle des Fensters, das von *CreateWindow* zurückgeliefert wird.

Klasse ist der Klassenname des Fensters, der im Ausschnitt *Klassenliste* angezeigt wird.

Modul ist der Name des ausführbaren Moduls (.EXE oder .DLL), der das Fenster erzeugt hat. Genauer gesagt ist es der Name desjenigen Moduls, der das Datensegment besitzt, das als Parameter *hInstance* an *CreateWindow* übergeben wurde.

Position ist »*hidden*« wenn das Fenster unsichtbar ist. Ist das Fenster sichtbar, so wird Position in Bildschirmkoordinaten (bei übergeordneten Fenstern) oder Koordinaten im Client-Bereich des übergeordneten Fensters (bei untergeordneten Fenstern) angegeben. Position wird in »olgendem Format angegeben:

```
(xBeginn, yBeginn)-(xEnde,yEnde)
```

Titel ist der Fenstertitel oder der Text, der von *GetWindowText* oder der Botschaft WM_GETTEXT zurückgeliefert wird. Wenn der Name ein leerer String ist, werden die Anführungszeichen weggelassen.

Ein Fenster finden

WinSight hat einen speziellen Modus, um Fenster zu lokalisieren. Er kann auf zweierlei Art und Weise arbeiten, entweder durch Identifizierung der Linie im Ausschnitt *Fensterhierarchie*, die dem Fenster entspricht, auf das Sie mit der Maus zeigen, oder durch Hervorheben eines Fensters, das Sie im Ausschnitt *Fensterhierarchie* auswählen.

Achtung! Alle anderen Anwendungsprogramme werden angehalten, während Sie im Modus *Fenster auswählen* arbeiten.

Gehen Sie in den Modus *Fenster suchen*, indem Sie *Überwachen/Fenster suchen* wählen. In diesem Modus erscheint immer dann, wenn Sie die Maus in eine Fensterbegrenzung hinein bewegen, ein dicker Rahmen um dieses Fenster, und es wird im Ausschnitt *Fensterhierarchie* selektiert.

Alternativ können Sie (wenn Sie sich im Modus *Fenster auswählen* befinden) ein Fenster im Ausschnitt *Fensterhierarchie* mit der Maus oder den Pfeiltasten auswählen. *WinSight* wird dann das oder die ausgewählten Fenster mit einem dicken Rahmen darstellen. Wenn Sie dann [*Eingabetaste*] betätigen, erscheint das Fenster *Fensterdetails* für das gewählte Fenster.

Den Modus Fenster auswählen *verlassen*

Wenn Sie das gewünschte Fenster lokalisiert haben, verlassen Sie den Modus *Fenster auswählen* per Mausklick oder durch Drücken der [*Esc*]-Taste. Dies löscht die dicke Umrandung auf dem Bildschirm, läßt aber die Beschreibung des aktuellen Fensters im Ausschnitt *Fensterhierarchie* angewählt.

Fenster überwachen

Um ein oder mehrere Fenster zu überwachen, selektieren Sie die Fenster (mit der Maus und der Taste [*Umschalt*] oder [*Strg*]), und wählen Sie dann *Botschaften/Ausgewählte Fenster*. Um andere als die zur Zeit überwachten Fenster zu überwachen, ändern Sie in *Fensterhierarchie* die Selektion.

Um alle Fenster zu überwachen, unabhängig davon, was in den Ausschnitten *Klassenliste* oder *Fensterhierarchie* ausgewählt wurde, wählen Sie *Botschaften/Alle Fenster*.

Nach Auswahl von *Botschaften/Ausgewählte Fenster* oder *Fenster/Alle Fenster* wird der Ausschnitt *Botschaften* sichtbar.

Die Auswahl von *Botschaften/Aufzeichnung aus* deaktiviert die Botschaftsverfolgung, ohne daß der Ausschnitt *Botschaften* unsichtbar wird.

Zu überwachende Botschaften filtern

Der Ausschnitt *Botschaften* zeigt die Botschaften an, die von ausgewählten Fensterklassen oder Fenstern empfangen werden. Die über *SendMessage* empfangenen Botschaften werden zweimal angezeigt: einmal, wenn sie gesendet werden und noch einmal, wenn sie zurückkehren, um den Rückgabewert anzuzeigen. *Dispatch*-Botschaften werden nur einmal gezeigt, da ihr Rückgabewert bedeutungslos ist. Die Botschaftsanzeige dient zur Darstellung der Verschachtelung von Botschaften innerhalb anderer Botschaften.

Mit dem Ausschnitt *Botschaften* arbeiten

Standardmäßig verfolgt *WinSight* alle Botschaften und zeigt sie im Ausschnitt *Botschaften* an. *WinSight* bietet Ihnen viele Möglichkeiten, um die Verfolgung von Botschaften auf bestimmte zu begrenzen.

- Wählen Sie *Botschaften/Ausgewählte Klassen* oder *Botschaften/Ausgewählte Fenster*, und selektieren Sie dann die Klassen (im Ausschnitt *Klassenliste*) bzw. Fenster (im Ausschnitt *Fensterhierarchie*) mit der Maus und der Taste [*Umschalt*] oder [*Strg*].
- Wählen Sie *Botschaften/Alle Fenster*.
- Wählen Sie *Botschaften/Optionen*, und selektieren Sie dann eine oder alle der 14 Botschaftsgruppen (die Gruppen sind in den Tabellen 7.1 bis 7.14 beschrieben). Aktivieren Sie *Alle Botschaften* im Dialogfenster *Optionen*, wenn Sie wieder alle Botschaften verfolgen wollen.

Das Dialogfenster *Botschaften/Optionen* erlaubt es Ihnen, das Format der Botschaften im Ausschnitt *Botschaften* zu verändern. Außerdem können Sie dort die Ausgabe der Botschaften in eine Datei, zu einem Drucker, zu einem zweiten Monitor oder in ein Hilfsfenster umleiten.

- Normalerweise interpretiert der Ausschnitt *Botschaften* jeden Botschaftsparameter und zeigt ihn in einem lesbaren Format an (falls das Markierungsfeld *Werte interpretieren* aktiviert ist). Bei Aktivierung von *Nur Hex* werden die Botschaftsparameter als Hexadezimalwerte von *wParam* und *lParam* dargestellt.
- Informationen zu verfolgten Botschaften werden normalerweise nur im Ausschnitt *Botschaften* angezeigt. Sie können die Ausgabe jedoch auch in eine Datei, zu einem Drucker oder zu einem zweiten Monitor umleiten, indem Sie *Logdatei* im Dialogfenster *Optionen für Botschaften* aktivieren und dann eins der folgenden Dinge tun:

Geben Sie einen Dateinamen ein, um die Botschaften in eine Protokolldatei schreiben zu lassen. Falls die Datei schon existiert, werden neue Daten am Ende angehängt.

Geben Sie den Namen des Geräts (z.B. PRN) ein, um die Ausgabe zu einer Druckerschnittstelle senden zu lassen.

Geben Sie AUX ein, um die Botschaften zu einem zweiten Monitor oder einem Fenster senden zu lassen. Hierzu müssen Sie über Ihre Datei CONFIG.SYS den Gerätetreiber WINOX.SYS oder OX.SYS installiert haben.

Um das Senden der Botschaften in eine Datei, zu einem Drucker oder zu einem zweiten Monitor zu beenden, deaktivieren Sie *LogDatei*.

Format `Handle ["Titel" oder {Klasse}] Botschaft Status`

Handle ist der Handle des Fensters, das die Botschaft empfängt.

Titel ist der Titel des Fensters. Wenn der Titel ein leerer String ist, wird stattdessen der Klassenname in geschweiften Klammern angezeigt.

Botschaft ist der Name der Botschaft, wie er in Windows definiert ist. Solche Namen werden von *WinSight* in Großbuchstaben angezeigt. Bekannte undokumentierte Windows-Botschaften werden in Kleinbuchstaben dargestellt. Unbekannte (benutzerdefinierte) Botschaftsnummern werden als `wm_User+0xXXXX` angezeigt, falls sie größer oder gleich *wm_User* sind, oder als `wm_0xXXXX`, falls sie kleiner als *wm_User* sind. Registrierte Botschaftsnummern (von *RegisterWindowsMessage)* werden mit ihrem registrierten Namen in einfachen Anführungsstrichen angezeigt.

Status ist einer oder mehrere der folgenden Werte:

- *Dispatched* zeigt an, daß die Botschaft über *DispatchMessage* empfangen wurde.
- *Sent [from XXXX]* zeigt an, daß die Botschaft über *SendMessage* empfangen wurde. Falls sie von einem anderen Fenster gesendet wurde, ist `from XXXX` das Handle des Fensters. Wurde sie vom gleichen Fenster ge-

sendet, von dem sie auch empfangen wurde, wird `from self` angezeigt. Falls sie von Windows gesendet wurde, entfällt die *from*-Angabe.

- *Returns* zeigt an, daß die Botschaft über *SendMessage* empfangen wurde und nun zurückkehrt.
- Außerdem können Botschaften Zahlenwerte oder Texte, wie etwa bei *wm_GetText*, enthalten. Für gesendete und *Dispatch*-Botschaften interpretiert *WinSight* die Parameter und zeigt sie in lesbarem Format an. Bei Botschaften, die mit Datenstrukturen verbunden sind (z.B. *wm_Create*), greift *WinSight* auf diese Daten zu und zeigt sie ebenfalls an.

Tabelle 7.1: Mausbotschaften

WM_HSCROLL	WM_MBUTTONUP	WM_RBUTTONDOWN
WM_LBUTTONDBLCLK	WM_MOUSEACTIVATE	WM_RBUTTONUP
WM_LBUTTONDOWN	WM_MOUSEFIRST	WM_SETCURSOR
WM_LBUTTONUP	WM_MOUSELAST	WM_VSCROLL
WM_MBUTTONDBLCLK	WM_MOUSEMOVE	
WM_MBUTTONDOWN	WM_RBUTTONDBLCLK	

Tabelle 7.2: Fensterbotschaften

WM_ACTIVATE	WM_GETDLGCODE	WM_QUERYNEWPALETTE
WM_ACTIVATEAPP	WM_GETFONT	WM_QUERYOPEN
WM_CANCELMODE	WM_GETMINMAXINFO	WM_QUIT
WM_CHILDACTIVATE	WM_GETTEXT	WM_SETFOCUS
WM_CLOSE	WM_GETTEXTLENGTH	WM_SETFONT
WM_CREATE	WM_ICONERASEBKGND	WM_SETREDRAW
WM_CTLCOLOR	WM_KILLFOCUS	WM_SETTEXT
WM_DDE_FIRST	WM_MOVE	WM_SHOWWINDOW
WM_DESTROY	WM_PAINT	WM_SIZE
WM_ENABLE	WM_PAINTICON	WM_WINDOWPOSCHANGED
WM_ENDSESSION	WM_QUERYDRAGICON	WM_WINDOWPOSCHANGING
WM_ERASEBKGND	WM_QUERYENDSESSION	

Tabelle 7.3: Eingabebotschaften

WM_CHAR	WM_KEYUP	WM_SYSKEYDOWN
WM_CHARTOITEM	WM_MENUCHAR	WM_SYSKEYUP
WM_COMMAND	WM_MENUSELECT	WM_TIMER
WM_DEADCHAR	WM_PARENTNOTIFY	WM_VKEYTOITEM
WM_KEYDOWN	WM_SYSCHAR	
WM_KEYLAST	WM_SYSDEADCHAR	

Tabelle 7.4: Systembotschaften

WM_COMPACTING	WM_PALETTECHANGED	WM_SYSCOLORCHANGE
WM_DEVMODECHANGE	WM_PALETTEISCHANGING	WM_SYSCOMMAND
WM_ENTERIDLE	WM_POWER	WM_TIMECHANGE
WM_FONTCHANGE	WM_QUEUESYNCH	WM_WININICHANGE
WM_NULL	WM_SPOOLERSTATUS	

Tabelle 7.5: Initialisierungsbotschaften

WM_INITDIALOG	WM_INITMENU	WM_INITMENUPOPUP

Tabelle 7.6: Botschaften der Zwischenablage

WM_ASKCBFORMATNAME	WM_DESTROYCLIPBOARD	WM_RENDERALLFORMATS
WM_CHANGECBCHAIN	WM_DRAWCLIPBOARD	WM_RENDERFORMAT
WM_CLEAR	WM_HSCROLLCLIPBOARD	WM_SIZECLIPBOARD
WM_COPY	WM_PAINTCLIPBOARD	WM_UNDO
WM_CUT	WM_PASTE	WM_VSCROLLCLIPBOARD

Tabelle 7.7: Botschaften für dynamischen Datenaustausch (DDE)

WM_DDE_ACK	WM_DDE_EXECUTE	WM_DDE_REQUEST
WM_DDE_ADVISE	WM_DDE_INITIATE	WM_DDE_TERMINATE
WM_DDE_DATA	WM_DDE_POKE	WM_DDE_UNADVISE

Tabelle 7.8: Non-Client-Botschaften

WM_NCACTIVATE	WM_NCLBUTTONDOWN	WM_NCPAINT
WM_NCCALCSIZE	WM_NCLBUTTONUP	WM_NCRBUTTONDBLCLK
WM_NCCREATE	WM_NCMBUTTONDBLCLK	WM_NCRBUTTONDOWN
WM_NCDESTROY	WM_NCMBUTTONDOWN	WM_NCRBUTTONUP
WM_NCHITTEST	WM_NCMBUTTONUP	
WM_NCLBUTTONDBLCLK	WM_NCMOUSEMOVE	

Tabelle 7.9: Druckerbotschaften

DM_COLOR	DM_MODIFY	DM_PRINTQUALITY
DM_COPIES	DM_ORIENTATION	DM_PROMPT
DM_COPY	DM_OUT_BUFFER	DM_SCALE
DM_DEFAULTSOURCE	DM_OUT_DEFAULT	DM_SPECVERSION
DM_DUPLEX	DM_PAPERLENGTH	DM_TTOPTION
DM_IN_BUFFER	DM_PAPERSIZE	DM_UPDATE
DM_IN_PROMPT	DM_PAPERWIDTH	DM_YRESOLUTION

Tabelle 7.10: Steuerbotschaften

BM_GETCHECK
BM_GETSTATE
BM_SETCHECK
BM_SETSTATE
BM_SETSTYLE

BN_CLICKED
BN_DISABLE
BN_DOUBLECLICKED
BN_HILITE
BN_PAINT
BN_UNHILITE

CB_ADDSTRING
CB_DELETESTRING
CB_DIR
CB_FINDSTRING
CB_FINDSTRINGEXACT
CB_GETCOUNT
CB_GETCURSEL
CB_GETDROPPEDCONTROLRECT
CB_GETDROPPEDSTATE
CB_GETEDITSEL
CB_GETITEMDATA
CB_GETEXTENDEDUI
CB_GETITEMDATA
CB_GETITEMHEIGHT
CB_GETLBTEXT
CB_GETLBTEXTLEN
CB_INSERTSTRING
CB_LIMITTEXT
CB_MSGMAX
CB_RESETCONTENT
CB_SELECTSTRING
CB_SETCURSEL
CB_SETEDITSEL
CB_SETITEMDATA
CB_SETITEMHEIGHT
CB_SHOWDROPDOWN

CBN_CLOSEUP
CBN_DBLCLK
CBN_DROPDOWN
CBN_EDITCHANGE
CBN_EDITUPDATE
CBN_ERRSPACE
CBN_KILLFOCUS
CBN_SELCHANGE
CBN_SELENDCANCEL
CBN_SETFOCUS

DM_GETDEFID
DM_SETDEFID

EM_CANUNDO
EM_EMPTYUNDOBUFFER
EM_FMTLINES
EM_GETFIRSTVISIBLELINE
EM_GETHANDLE
EM_GETLINE
EM_GETLINECOUNT
EM_GETMODIFY
EM_GETPASSWORDCHAR
EM_GETRECT
EM_GETSEL
EM_GETTHUMB
EM_GETWORDBREAKPROC
EM_LIMITTEXT
EM_LINEFROMCHAR
EM_LINEINDEX
EM_LINELENGTH
EM_LINESCROLL
EM_MSGMAX
EM_REPLACESEL
EM_SCROLL
EM_SETFONT
EM_SETHANDLE
EM_SETMODIFY
EM_SETPASSWORDCHAR
EM_SETRECT
EM_SETRECTNP
EM_SETSEL
EM_SETTABSTOPS
EM_SETWORDBREAK
EM_UNDO

EN_CHANGE
EN_ERRSPACE
EN_HSCROLL
EN_KILLFOCUS
EN_MAXTEXT
EN_SETFOCUS
EN_UPDATE
EN_VSCROLL

LB_ADDSTRING
LB_DELETESTRING
LB_DIR
LB_FINDSTRING
LB_FINDSTRINGEXACT
LB_GETCARETINDEX
LB_GETCOUNT
LB_GETCURSEL
LB_GETHORIZONTALEXTENT
LB_GETITEMDATA
LB_GETITEMHEIGHT
LB_GETITEMRECT
LB_GETSEL
LB_GETSELCOUNT
LB_GETSELITEMS
LB_GETTEXT
LB_GETTEXTLEN
LB_GETTOPINDEX
LB_INSERTSTRING
LB_MSGMAX
LB_RESETCONTENT
LB_SELECTSTRING
LB_SELITEMRANGE
LB_SETCARETINDEX
LB_SETCOLUMNWIDTH
LB_SETCURSEL
LB_SETHORIZONTALEXTENT
LB_SETITEMDATA
LB_SETITEMHEIGHT
LB_SETSEL
LB_SETTABSTOPS
LB_SETTOPINDEX

LBN_DBLCLK
LBN_ERRSPACE
LBN_KILLFOCUS
LBN_SELCANCEL
LBN_SELCHANGE
LBN_SETFOCUS

STM_GETICON
STM_SETICON

Tabelle 7.11: Stiftbotschaften

WIN_USER	WM_HOOKRCRESULT	WM_RCRESULT
WM_GLOBALRCCHANGE	WM_PENWINFIRST	WM_SKB
WM_HEDITCTL	WM_PENWINLAST	

Tabelle 7.12: Multimedia-Botschaften

MM_ADLIB	MM_MIM_CLOSE	MM_SNDBLST_MIDIIN
MM_JOY1_BUTTONDOWN	MM_MIM_DATA	MM_SNDBLST_MIDIOUT
MM_JOY1BUTTONUP	MM_MIM_ERROR	MM_SNDBLST_SYNTH
MM_JOY1MOVE	MM_MIM_LONGDATA	MM_SNDBLST_WAVEIN
MM_JOY1ZMOVE	MM_MIM_LONGERROR	MM_SNDBLST_WAVEOUT
MM_JOY2_BUTTONDOWN	MM_MIM_OPEN	MM_WAVE_MAPPER
MM_JOY2BUTTONUP	MM_MOM_CLOSE	MM_WIM_CLOSE
MM_JOY2MOVE	MM_MOM_DONE	MM_WIM_DATA
MM_JOY2ZMOVE	MM_MOM_OPEN	MM_WIM_OPEN
MM_MCINOTIFY	MM_MPU401_MIDIIN	MM_WOM_CLOSE
MM_MICROSOFT	MM_MPU401_MIDIOUT	MMWOM_DONE
MM_MIDI_MAPPER	MM_PC_JOYSTICK	MM_WOM_OPEN

Tabelle 7.13: Sonstige Botschaften

WM_COALESCE_FIRST	WM_MDIACTIVATE	WM_MDIRESTORE
WM_COALESCE_LAST	WM_MDICASCADE	WM_MDISETMENU
WM_COMMNOTIFY	WM_MDICREATE	WM_MDITILE
WM_COMPAREITEM	WM_MDIDESTROY	WM_MEASUREITEM
WM_DELETEITEM	WM_MDIGETACTIVE	WM_NEXTDLGCTL
WM_DRAWITEM	WM_MDIICONARRANGE	WM_SYSTEMERROR
WM_DROPFILES	WM_MDIMAXIMIZE	
WM_KEYFIRST	WM_MDINEXT	

Tabelle 7.14: Von Microsoft nicht dokumentierte Botschaften

WM_ALTTABACTIVE	WM_ENTERSIZEMOVE	WM_QUERYPARKICON
WM_BEGINDRAG	WM_EXITMENULOOP	WM_SETHOTKEY
WM_CONVERTREQUEST	WM_EXITSIZEMOVE	WM_SETVISIBLE
WM_CONVERTRESULT	WM_FILESYSCHANGE	WM_SIZEWAIT
WM_DRAGLOOP	WM_GETHOTKEY	WM_SYNCPAINT
WM_DRAGMOVE	WM_ISACTIVEICON	WM_SYNCTASK
WM_DRAGSELECT	WM_LBTRACKPOINT	WM_SYSTIMER
WM_DROPOBJECT	WM_NEXTMENU	WM_TESTING
WM_ENTERMENULOOP	WM_QUERYDROPOBJECT	WM_YOMICHAR

T E I L II

Verwendung der Kommandozeilen-Hilfsprogramme

Teil II dieses Handbuchs behandelt die mit Borland C++ gelieferten Kommandozeilen-Hilfsprogramme. Diese umfassen Compiler, Linker, einen Ressourcen-Compiler, eine Bibliotheksverwaltung, einen Projekterzeuger (genannt MAKE) und andere Hilfsprogramme. Erläuterungen zu den meisten dieser Programme finden Sie in diesem Teil, Beschreibungen zu einigen anderen in Online-Dateien. Zusätzlich findet sich eine Dokumentation sämtlicher Hilfsprogramme in der Online-Hilfe.

Während des Programmierens können Sie entweder die IDE oder die Kommandozeilen-Hilfsprogramme verwenden. Obwohl Sie mit beiden die gleichen Ergebnisse erzielen, werden Sie vielleicht die Kommandozeilen-Hilfsprogramme vorziehen, weil Sie lieber Ihren eigenen Programmeditor (z.B. Brief) benutzen möchten. Es folgt eine Auflistung der Kommandozeilen-Hilfsprogramme, eine Beschreibung ihrer Funktionen und der Stellen, an denen sie dokumentiert sind.

- Kapitel 7 beschreibt das Hilfsprogramm WinSight. Obwohl dieses, technisch gesehen, kein Kommandozeilen-Hilfsprogramm ist, wird es hier als Programmierwerkzeug aufgeführt.
- Kapitel 8 beschreibt das Hilfsprogramm WinSpector. Obwohl dieses, technisch gesehen, kein Kommandozeilen-Hilfsprogramm ist, wird es hier als Programmierwerkzeug aufgeführt.
- Kapitel 9 enthält eine Beschreibung von BCC.EXE, den 16- und 32-Bit-Kommandozeilen-Compilern von Borland C++, und eine Beschreibung des Borland C++ Linkers TLINK.EXE. Dieser Linker erzeugt .EXE- und .DLL-Dateien unter Verwendung von .OBJ- und .LIB-Dateien.

- Kapitel 10 beschreibt die Ressourcen-Hilfsprogramme BRCC.EXE, BRC.EXE und RLINK.EXE, die Anwendungsressourcen compilieren.
- Kapitel 11 beschreibt die Hilfsprogramme IMPLIB.EXE und TLIB.EXE, mit denen Sie C und C++ Bibliotheken erstellen und pflegen können.
- Kapitel 12 beschreibt die Borland C++ Make-Utilities MAKE.EXE und MAKER.EXE. Diese Hilfsprogramme helfen Ihnen bei der Verwaltung Ihrer Projekte, indem sie nur die Dateien compilieren, die sich seit dem letzten Compilieren geändert haben.
- HC31.EXE compiliert Dateien für Online-Hilfeprogramme und erzeugt .HLP-Dateien in einer Form, die in den meisten Windows-Anwendungen verwendet werden kann. Dieses Programm ist in der Online-Hilfe dokumentiert.

Für weitere Informationen vergleichen Sie bitte die Online-Hilfe und die Readmes.

Ausführen der Kommandozeilen-Hilfsprogramme

Der Kommandozeilen-Compiler verwendet DPMI (DOS Protected Mode Interface, DOS-Schnittstelle für den Protected Mode), um auf 386er-, i486er- und Pentium-Rechnern mit mindestens 640KB konventionellem RAM und mindestens 4MB Erweiterungsspeicher im Protected Mode zu laufen.

Obwohl die Compiler im Protected Mode laufen, erzeugen Sie Anwendungen, die im Real Mode laufen. Protected Mode-Hilfsprogramme haben den Vorteil, daß sie auf mehr Speicher zugreifen können als Real Mode-Hilfsprogramme. So können Sie umfangreiche Projekte mit größerer Geschwindigkeit compilieren, ohne ausgedehnte Datenauslagerungen vornehmen zu müssen.

Speicher und MAKESWAP.EXE

Wenn Sie die 32-Bit-Kommandozeilen-Hilfsprogramme ausführen und von DOS die Fehlermeldung »Nicht genügend Speicherplatz« erhalten, erstellen Sie eine Swap-Datei mit dem Hilfsprogramm MAKESWAP. Geben Sie die Größe der zu erzeugenden Datei in KB an. Folgende Syntax wird ebenfalls unterstützt:

```
MAKESWAP 12M
```

Der gezeigte Befehl erzeugt eine 12MB große Swap-Datei namens EDPMI.SWP im aktuellen Verzeichnis, die dann die Kommandozeilen-Hilfsprogramme verwenden, wenn sie zusätzlichen Speicher benötigen. Um eine Swap-Datei einzurichten, verwenden Sie die Umgebungsvariable DPIMEM an der DOS-Eingabeaufforderung oder fügen Sie Ihrer AUTOEXEC.BAT-Datei folgende Zeile hinzu:

```
set DPMI32=SWAPFILE <Pfadname der Swap-Datei>EDPMI.SWP
```

Hinweis

MAKESWAP ist nur unter DOS verwendbar, nicht in unter Windows geöffneten DOS-Fenstern. In der Online-Datei INSTALL.TXT finden Sie weitere Informationen zum Starten der Hilfsprogramme von DOS-Fenstern aus.

Der Laufzeit-Manager und Hilfsprogramme

Die Borland C++ Protected Mode-Anwendungen (z.B. BCC und BCC32) verwenden die Laufzeit-Manager RTM.EXE und 32RTM.EXE. Die den Laufzeit-Manager benutzenden Hilfsprogramme laden diesen zunächst, führen ihre Aufgaben durch und beenden den Laufzeit-Manager wieder. Wenn Sie auf 32-Bit-Kommandozeilen-Hilfsprogramme zugreifen, die den Laufzeit-Manager mehrmals in kurzen Abständen nacheinander verwenden (vielleicht von einer Make-Datei aus), können Sie den Prozess beschleunigen, indem Sie den Laufzeit-Manager nur einmal laden und dann die Hilfsprogramme aufrufen. Um den Laufzeit-Manager zu laden, geben Sie in der Befehlszeile `32RTM` ein; geben Sie `32RTM -u` ein, um ihn zu beenden.

Der Laufzeit-Manager nimmt beim Laden standardmäßig den gesamten verfügbaren Speicher ein. Er reserviert dann Speicher für die Hilfsprogramme, wenn diese den Speicher über die API-Routinen der Speicherverwaltung anfordern.

Wenn der Laufzeit-Manager unter Windows 3.x in einem DOS-Fenster läuft, wird die Größe des von ihm reservierten Speichers auch beschränkt durch den KB-Grenzwert für XMS-Speicher in der entsprechenden .PIF-Datei. Die Einstellung in Ihrer .PIF-Datei für das DOS-Fenster sollte diesen Grenzwert auf mindestens 1024KB setzen. Er setzt die Grenze für die Speichergröße, die der Laufzeit-Manager für DOS-Anwendungen mit Nutzung von Erweiterungsspeicher verwendet.

KAPITEL 8

WinSpector

WinSpector und die zugehörigen Dienstprogramme unterstützen Sie nach dem Auftreten von UAEs (Unrecoverable Application Error = Nicht behebbarer Fehler in Anwendungsprogramm) und GDFs (General Protection Error = Allgemeine Schutzverletzung) bei der Suche nach der Ursache. Wenn ein UAE oder ein GDF auftritt, schreibt WinSpector eine Protokolldatei auf die Platte, die für Sie nützliche Informationen über den Grund der Exception enthält, nämlich:

- Aufruf-Stack, der aktiv war, als die Exception auftrat.
- Funktions- und Prozedurnamen im Aufruf-Stack
- CPU-Register
- Disassemblierte Maschinenbefehle am Ort der Exception
- Windows-Informationen über die Programmumgebung

Mit WinSpector arbeiten

Stellen Sie vor der Arbeit mit WinSpector sicher, daß sich *TOOLHELP.DLL* (Version aus Windows 3.1 oder später) in Ihrem Suchpfad befindet. TOOLHELP.DLL wird bei Borland C++ mitgeliefert. TOOLHELP.DLL ist eine Windows-DLL, die beliebigen Dienstprogrammen Zugriff auf Systeminformationen der unteren Ebenen ermöglicht. WinSpector nutzt TOOLHELP.DLL zum Protokollieren von Exceptions und erhält von dort Systeminformationen, die es in die Protokolldatei schreibt. Während WinSpector läuft, sollte außer dem Turbo Debugger kein anderes Debug-Dienstprogramm, das Exceptions verarbeitet, benutzt werden.

Sie können WinSpector (hat nach dem Laden Symbolgröße) auf drei Arten starten:



- Nehmen Sie WinSpector in den Abschnitt »load=« der Datei WIN.INI auf.
- Nehmen Sie WinSpector in die Autostart-Gruppe von Windows auf.

- Doppelklicken Sie nach dem Starten von Windows auf das WinSpector-Symbol, um WinSpector zu starten.

Wenn eine Exception (UAE oder GPF) auftritt, schreibt WinSpector in eine Datei mit dem Namen WINSPCTR.LOG (eine Textdatei) einen Bericht Meldung, die zur Ermittlung der Ursache der Exception nützliche Informationen enthält. Außerdem erzeugt WinSpector die Datei WINSPCTR.BIN; dies ist eine Binärdatei, aus der das Dienstprogramm DFA die Textdatei DFA.OUT erzeugen kann.

Nach dem Auftreten einer Exception zeigt WinSpector ein Dialogfenster mit einer kurzen Meldung, die Exception betreffend, an. Klicken Sie dann auf OK, um das Fenster zu schließen, und lesen Sie den Inhalt der Protokolldatei, um die Ursache der Exception zu ermitteln. Im folgenden Abschnitt wird beschrieben, wie Sie festlegen können, was in die Datei WINSPCTR.LOG geschrieben wird.

WINSPCTR.LOG konfigurieren

Es gibt zwei Wege, um die Optionen für WinSpector zu spezifizieren, die die Ausgabe in WINSPCTR.LOG bestimmen:

- Per Dialogfenster *Vorgaben*: Starten Sie WinSpector, klicken Sie auf sein Symbol, und wählen Sie *Vorgaben* aus dem Popup-Menü.
- Per Datei WINSPCTR.INI: Laden Sie die Datei mit irgendeinem Texteditor, ändern oder ergänzen Sie die Befehle für WinSpector, speichern Sie die Datei ab, und starten Sie WinSpector neu.

Die folgenden Abschnitte beschreiben jede Option im Dialogfenster *Vorgaben*. Am linken Rand sind die Optionen für WINSPCTR.INI angegeben.

LogDir=[Verzeichnis]

- Die Option *Verzeichnis* legt fest, wo die Datei WINSPCTR.LOG gespeichert wird. Geben Sie hier den gewünschten Pfad ein (Voreinstellung ist C:\WINDOWS).

LogViewer=[Texteditor_Name]

- Die Option Texteditor legt fest, welches Programm WinSpector zur Anzeige der Protokolldatei benutzen soll. Geben Sie hier den Pfad- und Dateinamen für den von Ihnen gewünschten Texteditor ein (Voreinstellung ist NOTEPAD.EXE). Beispiel für eine Eingabe: `C:\WIN31\WRITE.EXE`. Falls WinSpector den Editor nicht findet, zeigt er die Meldung

```
Error: Unable to execute: [option]
```

an. Dabei steht *option* für den Dateinamen des Editors. Stellen Sie sicher, daß sich der angegebene Editor im angegebenen Verzeichnis befindet.

CreateNewLog= 0 (Anhängen) oder 1 (Überschreiben)

- Mit den Optionen *Neue Berichte anhängen* und *Vorige Berichte überschreiben* können Sie festlegen, ob WinSpector einen neu erstellten Report an die existierende Protokolldatei anhängt oder ob letztere überschrieben wird, wenn ein neuer Report erzeugt wird.

ShowSystemInfo= 0 (Weglassen) oder 1 (Aufnehmen)

- Wählen Sie die Option *System-Informationen*, um die Task-Liste, die Modulliste und Informationen über die USER- und GDI-Heaps in die Protokolldatei aufzunehmen.

LogToStdAux= 0 (Nein) oder 1 (Ja)

- Wählen Sie die Option *Zusammenfassung nach AUX*, wenn eine verkürzte Zusammenfassung der in die Protokolldatei geschriebenen Informationen auch zum Gerät AUX gesendet werden soll. Um diese Möglichkeit zu nutzen, benötigen Sie ein an AUX angeschlossenes Terminal oder einen Gerätetreiber (wie zum Beispiel OX.SYS), der die AUX-Ausgaben auf einen zweiten Bildschirm umlenkt.

PostMortemDump= 1 (Ausgeben) oder 0 (Weglassen)

- Wählen Sie die Option *Post-Mortem-Dump*, wenn eine WINSPCTR.BIN-Datei erzeugt werden soll. Das Programm DFA.EXE kann diese BIN-Datei in eine für Sie lesbare Textdatei übersetzen.

ShowStackInfo = 1 (Aufnehmen) oder 0 (Weglassen)

- Wählen Sie die Option *Stack-Frame Daten*, wenn WinSpector ausführliche Stack-Trace-Informationen in die Protokolldatei aufnehmen soll. Für jeden Stack-Frame, der nicht größer als 256 Byte ist, wird ein Speicherauszug erstellt, der bei der Adresse SS:BP dieses Frames beginnt. Falls zwischen zwei aufeinanderfolgenden Stack-Frames mehr als 256 Byte liegen, entfällt die Ausgabe für diesen Frame. Diese Daten können Sie verwenden, um die Werte der an die Funktion übergebenen Parameter zu ermitteln.

Es ist gewöhnlich einfacher, wenn Sie die schwierige Aufgabe, die Parameter zu analysieren, dem Dienstprogramm DFA überlassen. Falls jedoch keine Turbo-Debugger-Informationen zur Verfügung stehen, kann Ihnen ein solcher ausführlicher Trace hilfreich sein.

ShowUserInfo= 1 (Aufnehmen) oder 0 (Weglassen)

- Wählen Sie die Option *Kommentare*, wenn WinSpector von Ihnen selbst gelieferte Informationen über die Situation, in der die Exception auftrat, in die Protokolldatei schreiben soll. Ist diese Option aktiviert, so öffnet WinSpector unmittelbar nach dem Auftreten einer Exception ein Dialogfeld; der dort von Ihnen eingegebene Text wird am Ende des Reports in die Protokolldatei geschrieben.

WINSPCTR.LOG Referenz

Jeder Report in der Datei WINSPCTR.LOG besteht aus mehreren Abschnitten, deren Inhalt Ihnen hilft, die Ursache der Exception in Ihrem Programm zu ermitteln. Die erste Zeile jedes Reports in WINSPCTR.LOG enthält die Angabe von Datum und Uhrzeit beim Auftreten der Exception. Beispiel:

```
WinSpector failure report - 6/18/1992 11:04:25
```

Die zweite Zeile enthält folgende Angaben:

- Typ der Exception (Tabelle 8.1 zeigt die häufigsten Typen)
- Name des Moduls
- Logische Adresse
- Physische Adresse

- Aktuelle Task zum Zeitpunkt der Exception

Beispiel für eine solche zweite Zeile:

```
Exception 13 at USER 002A:0429 (079F:0429) (TASK=SCHLECHT)
```

Tabelle 8.1: Typen von Exceptions

Nummer	Bezeichnung	Beschreibung
0	Division durch null	Tritt während einer DIV- oder IDIV-Operation auf, wenn der Divisor 0 ist.
12	Stacküberlauf	Tritt normalerweise auf, wenn der für den Stack verfügbare Speicherplatz erschöpft ist.
13	Allgemeine Schutzverletzung (GPF)	Alle Schutzverletzungen, die unter keinen anderen Exception-Typ fallen, verursachen die Exception 13.

Unter die Exception-Nummer 13 fallen folgende (aber auch andere) Fehler:

- Laden eines unzulässigen Selektors in ein Segmentregister
- Überschreiten der Segmentgrenze, wobei der Selektor zwar zulässig ist, der Offsetwert aber die Segmentgrenze überschreitet. Beispiel: Eine Feldindexüberschreitung im DS-, ES- oder einem anderen Segment.
- Übergabe der Ablaufkontrolle an ein nichtausführbares Segment, zum Beispiel durch einen ungültigen Zeiger auf eine Funktion.
- Zugriff auf die Register DS, ES, FS oder GS, wenn diese einen NULL-Selektor enthalten. (Bei dieser Exception kann für den Segmentregisterinhalt in der Protokolldatei der Wert null erscheinen.)

In der Protokolldatei werden sowohl die physische als auch die logische Adresse des Orts der Exception angegeben. Bei Windows-Programmen sind diese beiden Arten von Adressen aus folgenden Gründen wichtig:

- Beim Laden eines Programms reserviert Windows Speicherplatz für jedes logische Segment und ordnet jedem Segment einen eindeutigen Selektor zu. Der Selektor und ein zugehöriger Offset repräsentieren zusammen eine physische Adresse.
- Beim Linken einer Windows-EXE-Datei wird jedes Segment in einen eigenen Abschnitt der Datei plaziert, und es wird eine Segmenttabelle angelegt.
- Eine logische Adresse, die in Wirklichkeit die Position eines Segments in der Segmenttabelle von Windows angibt, besteht aus einem Modulnamen, einem logischen Segment und einem Offset. Um Informationen über Segmentgrenzen und anderes zu erhalten, können Sie das Dienstprogramm TDUMP auf die Datei anwenden. Alternativ können Sie auch eine MAP-Datei erzeugen, sie enthält die gleichen Informationen.

Wenn der Stack-Zeiger im Moment der Exception einen zu kleinen Wert hat, schaltet TOOLHELP.DLL automatisch auf einen anderen Stack um

und schreibt die Meldung `Stack switched` ans Ende der zweiten Zeile der Protokolldatei.

Abschnitt Disassemblierung

Die erste Protokollzeile im Abschnitt *Disassemblierung* der Datei WINSPCTR.LOG gibt den Assemblerbefehl an, der die Exception verursacht hat. Dahinter kommen einige der nachfolgenden Programmbefehle, die es erleichtern, die fehlerverursachende Stelle in der Task zu finden.

Beispiel: Im folgenden Programmfragment enthält das Segmentregister ES einen Selektor und BX den Offset dieses Segments. Der Ausnahmezustand 13 tritt ein, falls der Inhalt von BX größer als die Segmentgrenze des durch ES bezeichneten Segments ist.

```
079F:0429       CMP                         BYTE PTR ES:[BX],FF
079F:042D       JNE                         043A
079F:042F       CMP                         WORD PTR [BP+06],03
079F:0435       MOV                         DI,0001
```

Abschnitt Stack-Trace

Die erste Zeile des Abschnitts *Stack-Trace* in der Protokolldatei WINSPCTR.LOG gibt die Funktion oder Prozedur an, die sich zum Zeitpunkt der Exception gerade in Ausführung befand. Die Stack-Trace-Informationen bestehen aus folgenden Angaben:

- Frame-Nummer
- Modulname
- Name derjenigen Funktion, die der die Exception verursachenden Funktion in Rückwärtsrichtung im Speicher am nächsten liegt; außerdem eine Zahl, die den Abstand zu dieser Funktion angibt. Diese Angaben erfolgen nur, wenn eine SYM-Datei verfügbar ist.
- Logische und physische Adresse des Stack-Frame.
- Rücksprungadresse Ihres Programms nach dem Aufruf.

Um einen Funktionsnamen angeben zu können, sucht WinSpector in der SYM-Datei nach demjenigen Symbolnamen, der von der Adresse her am dichtesten vor der Adresse im Aufruf-Stack liegt. Dazwischen können aber noch andere Funktionen liegen, da einige SYM-Dateien nicht über alle Funktionen Informationen enthalten. Deshalb kann der angegebene Funktionsname unzutreffend sein, besonders, wenn der angegebene Offset zu groß erscheint.

Der folgende Ausschnitt aus dem Abschnitt *Stack-Trace* zeigt einige der Funktionen, die ausgeführt wurden, als die Task SCHLECHT eine Exception verursachte:

```
Stack Trace:
0  User   <no info>
   CS:IP 002A:0429 (079F:0429)  SS:BP  10FF:18CA
   C:\WIN31\SYSTEM\USER.EXE
```

```
   :
3  SCHLECHT funktion5(unsigned long, unsigned long, unsigned long) + 0014
   CS:IP 0001:0184 (1107:0184)  SS:BP  10FF:1952
   \BIN\SCHLECHT.EXE
   :
```

Abschnitt Register

Im Abschnitt *Register* der Datei WINSPCTR.LOG werden die Inhalte der Standardregister zum Zeitpunkt der Exception wie im folgenden Beispiel angegeben:

```
Registers:
AX  0037
BX  0000
CX  0008
DX  10EE
SI  0037
DI  0028
```

Für die Register CS, DS, ES und SS werden Segmentgrenzen und Zugriffsrechte angegeben.

Abschnitt Botschaftswarteschlange

Im Abschnitt *Botschaftswarteschlange* der Datei WINSPCTR.LOG wird die während der Bearbeitung zuletzt empfangene Botschaft angegeben. Außerdem werden alle Botschaften angezeigt, die sich zum Zeitpunkt der Exception in der Warteschlange befanden. Zu jeder Botschaft macht WinSpector folgende Angaben:

Im Abschnitt *Botschaftswarteschlange* fehlt möglicherweise die letzte vom Programm empfangene Botschaft, weil Windows die Warteschlan-

- Fenster-Handle (bezeichnet das empfangende Fenster)
- Botschafts-ID-Nummer (bezeichnet die Botschaft)
- Zwei Parameter (zusätzliche Information über die Botschaft)

Das folgende Beispiel einer Botschaftswarteschlange zeigt eine bereits empfangene und eine wartende Botschaft:

```
Message Queue:
Last message received:
   hWnd: 0000   msg: 0001   wParam: 0002   lParam: 00000003
Waiting in queue:
   hWnd: 0000   msg: 0001   wParam: 0002   lParam: 00000003
```

Abschnitt Tasks

Im Abschnitt *Tasks* von WINSPCTR.LOG werden die Programme aufgeführt, die zum Zeitpunkt der Exception liefen. Für jedes Programm gibt es dabei folgende Angaben:

- Vollständiger Pfad der ausführbaren Datei
- Modulname
- Windows-Modul-Handle
- Task-Handle
- Wert des Datensegments der Task (Instanz-Handle)

Wenn die Task SCHLECHT eine Exception verursacht hat, werden beispielsweise unter anderem folgende Tasks angegeben:

```
C:\WIN31\SYSTEM\NWPOPUP.EXE
   Module: NWPOPUP    hModule: 142F   hTask: 141F   hInstance: 13F6
   :
C:\BIN\WINSPCTR.EXE
   Module: WINSPCTR   hModule: 1397   hTask: 1387   hInstance: 135E
   :
C:\BIN\SCHLECHT.EXE
   Module: SCHLECHT   hModule: 1467   hTask: 1127   hInstance: 10FE
```

Abschnitt Module

Im Abschnitt *Module* der Datei WINSPCTR.LOG werden die Module aufgeführt, die zum Zeitpunkt der Exception aktiv waren. Für jedes Modul gibt es folgende Angaben:

- Pfad der ausführbaren Datei
- Datumsangabe der ausführbaren Datei
- Größe der ausführbaren Datei
- Modulname
- Modul-Handle
- Zugriffszähler (gibt an, wie oft das Modul benutzt wird)

Wenn die Task SCHLECHT eine Exception verursacht hat, werden beispielsweise unter anderem folgende Tasks angegeben:

```
C:\WIN31\SYSTEM\KRNL386.EXE    Date: 03/02/1992    Size: 116132
   Module: KERNEL   hModule: 010F   reference count: 21
C:\WIN31\SYSTEM\SYSTEM.DRV     Date: 03/01/1992    Size: 2304
   Module: SYSTEM   hModule: 013F   reference count: 13
   :
C:\BIN\WINSPCTR.EXE            Date: 06/02/1992    Size: 46256
   Module: WINSPCTR hModule: 1397   reference count: 1
   :
```

Abschnitt USER- und GDI-Heap-Information

Der Abschnitt *USER- und GDI-Heap* (GDI: Graphics Device Interface = grafische Geräteschnittstelle) in WINSPCTR.LOG gibt an, wieviel Prozent des USER- und des GDI-Heap zum Zeitpunkt der Exception noch frei waren. Beispiel:

```
USER Free 91%
GDI Free 83%
```

Weil Windows den Anwendungsprogrammen nur einen gemeinsamen, 64 KByte großen internen Heap zur Verfügung stellt, ist es oft hilfreich, zu kontrollieren, wie stark dieser Speicherbereich genutzt wird. Wenn Sie bemerken, daß USER und GDI große Mengen des Heap-Speicherplatzes in Anspruch nehmen, sollten Sie überprüfen, ob Ihr Programm wirklich alle Ressourcen wieder freigibt, die es nicht mehr nutzt. Das Info-Fenster

des Windows-Programm-Managers zeigt den kleineren der beiden Werte als »Freie Systemressourcen« an.

Abschnitt Systeminformation

Der Abschnitt *Systeminformationen* in WINSPCTR.LOG liefert folgende Angaben über die Windows-Version und den aktuellen System-Modus:

- CPU-Typ
- Größter freier, linear zusammenhängender Speicherblock im System
- Gesamter linearer Adreßraum in Seiten
- Anzahl der freien Speicherseiten im linearen Adreßraum
- Anzahl der Seiten in der Auslagerungsdatei des Systems

Für ein 486er-System könnte der Abschnitt *Systeminformation* so aussehen:

```
System info: Running in enhanced mode under Windows 3.1 debug version
CPU: 80486
Largest free memory block: 3457024 bytes
Total linear memory space: 19696 K
Free linear memory space: 18212 K
Swap file pages: 0(0K)
```

Verarbeiten der Daten von WinSpector

Das Dienstprogramm DFA liest eine WINSPCTR.BIN-Datei und Informationen für den Turbo Debugger (enthalten in EXE-, DLL- oder TDS-Dateien) und wandelt die binären Daten in eine gut lesbare Form um. Zu diesem Zweck erzeugt DFA eine Datei, die außer Stack-Trace-Informationen (ähnlich denen in der Protokolldatei) auch Funktionsnamen, Zeilennummern sowie lokale und globale Variablen enthält.

DFA übernimmt die Nachbearbeitung der zum Zeitpunkt der Exception von WinSpector erfaßten Turbo-Debugger-Informationen. Wenn Sie die Option *Post-Mortem-Dump* aktivieren, legt WinSpector beim Auftreten einer Exception die Datei WINSPCTR.BIN an. Anschließend können Sie das Dienstprogramm DFA benutzen, um die in WINSPCTR.BIN enthaltenen Binärdaten in eine lesbare Textdatei mit dem Namen DFA.OUT umzuwandeln.

In jeder Arbeitssitzung mit Windows wird nur eine WINSPCTR.BIN-Datei angelegt, Sie sollten DFA deshalb unverzüglich aufrufen. Wenn zum Beispiel in einer Sitzung drei Exceptions nacheinander auftreten, schreibt WinSpector zwar drei Berichte in die Protokolldatei, aber es sind dann nur für den ersten Bericht Binärdaten vorhanden. Es empfiehlt sich also, das DFA-Dienstprogramm sofort nach der ersten Exception aufzurufen. Benennen Sie anschließend die Datei DFA.OUT um, und löschen Sie die Dateien WINSPCTR.BIN und WINSPCTR.LOG, bevor Sie weiterarbeiten.

DFA-Ausgabe

Das Dienstprogramm DFA legt nur dann eine Datei an, wenn Turbo-Debugger-Informationen für die im Stack-Frame angegebene Datei vorhanden sind. Die Stack-Trace-Informationen der DFA-Ausgabedatei (DFA.OUT) ähneln denen in der Protokolldatei von WinSpector, enthalten jedoch zusätzlich folgende Angaben:

- Funktionsnamen
- Zeilennummern
- Lokale und globale Variablen
- Datensegmente und ihre Werte (einschließlich des Stack-Segments)

DFA und WINSPCTR.LOG

Wird DFA allein mit der Datei WINSPCTR.LOG verwendet, so liefert es lediglich grundlegende Stack-Trace-Informationen wie beispielsweise Adressen. Falls Turbo-Debugger-Informationen (enthalten in einer EXE-, DLL- oder TDS-Datei) verfügbar sind, werden dem Report auch Quelltextdateinamen und Zeilennummern hinzugefügt.

DFA und WINSPCTR.BIN

Wird DFA zusammen mit der Datei WINSPCTR.BIN verwendet, so liefert es weitergehende Informationen:

- Im Protokoll erscheinen auch im Stack liegende Variablen einschließlich lokaler Variablen, an Funktionen übergebener Parameter, Strukturen und Arrays.
- Variablentypen, Werte und Adressen werden nach Funktionen getrennt aufgelistet.

Wenn Turbo-Debugger-Informationen vorhanden sind, so nimmt DFA für jeden Stack-Frame die folgenden Informationen in seine Reports auf:

- In Abschnitt 1:
 - Quelltextdatei
 - Lokale Variablen
 - Zeilennummer
 - Parameter
- In Abschnitt 2:
 - Modulname der fehlerhaften Task
 - Dateinamen
 - Logische Segmente
 - Segmentselektoren
 - Kennzeichnung: Daten- oder Code-Segment
- In Abschnitt 3:
 - Globale Variablen
 - Statische Variablen
 - Werte der Variablen zum Zeitpunkt der Exception

Syntax

```
DFA [Optionen] WINSPCTR.LOG [WINSPCTR.BIN]
```

Die Angabe von WINSPCTR.LOG ist erforderlich, mit dieser Datei erhalten Sie Informationen über Quelltextdatei und Zeilennummern. Ist auch WINSPCTR.BIN (optional) vorhanden, so kommen noch Informationen über die Variablen hinzu.

Tabelle 8.2: Optionen für DFA

Option	Beschreibung
/O[Ausgabedatei]	Veranlaßt DFA, für die Ausgabe einen von der Voreinstellung DFA.OUT abweichenden Namen zu verwenden.
/D	Veranlaßt DFA, einen hexadezimalen Speicherabzug der geretteten Datensegmente auszugeben.

Weitere Dienstprogramme von WinSpector

Zu WinSpector gehören drei Dienstprogramme, die noch weitergehende Informationen über eine Exception liefern:

- EXEMAP.EXE erzeugt eine MAP-Datei aus einer Windows-EXE-Datei. Diese MAP-Datei wird zur Erzeugung einer SYM-Datei benötigt, die wiederum einen erweiterten Fehlerreport für die zugrundeliegenden EXE-Datei ermöglicht.
- TMAPSYM.EXE wird in Verbindung mit EXEMAP.EXE eingesetzt und erzeugt eine SYM-Datei aus einer EXE-Datei.
- BUILDSYM.EXE stützt sich auf EXEMAP.EXE und TMAPSYM.EXE und erzeugt eine SYM-Datei aus einer Windows-EXE-Datei.

Arbeiten mit EXEMAP.EXE

Das Dienstprogramm EXEMAP erzeugt MAP-Dateien aus ausführbaren Windows-Dateien. Solche MAP-Dateien können bei der Erzeugung von SYM-Dateien verwendet werden, die wiederum von WinSpector genutzt werden, um seine Fehlerreports mit zusätzlichen Informationen auszustatten. Diese Informationen sind besonders dann nützlich, wenn Sie mit DLLs oder anderen Programmen arbeiten, deren Quelltext Ihnen nicht zur Verfügung steht.

Geben Sie `EXEMAP DateiName.EXE NeuerName.MAP` ein, um eine MAP-Datei aus einer EXE-Datei zu erzeugen. Wenn Sie keinen neuen Namen angeben, legt EXEMAP eine MAP-Datei mit demselben Namen (aber der Erweiterung MAP) wie die EXE-Datei an.

Die erzeugte MAP-Datei enthält zwar nicht so viele Informationen wie eine vom Linker im Übersetzungsprozeß erzeugte, aber sie enthält doch wenigstens die Adressen der exportierten öffentlichen Funktionen.

Arbeiten mit TMAPSYM.EXE

Das Dienstprogramm TMAPSYM erzeugt SYM-Dateien aus vorhandenen MAP-Dateien, die ihrerseits vom Compiler oder vom Dienstprogramm EXEMAP erzeugt worden sind. Aus diesen SYM-Dateien kann WinSpector Angaben über öffentliche Funktionen, Namen von Variablen und Funktionen in der Einsprungtabelle des ausführbaren Programms entnehmen. Konstanten und Zeilennummern sind allerdings in einer von TMAPSYM erzeugten SYM-Datei nicht enthalten.

Geben Sie `TMAPSYM DateiName.MAP` ein, um eine SYM-Datei aus einer MAP-Datei zu erzeugen. (Die Angabe der Namenserweiterung MAP ist erforderlich.)

Arbeiten mit BUILDSYM.EXE

Das Dienstprogramm BUILDSYM erzeugt SYM-Dateien aus EXE-Dateien. Es liefert dieselbe Ausgabe wie die Kombination von EXEMAP und TMAPSYM, da es diese Programme automatisch aufruft; im Unterschied dazu löscht es aber zum Schluß die MAP-Datei in Ihrem Verzeichnis wieder. BUILDSYM unterstützt auch Jokerzeichen, so daß Sie durch Eingabe eines einzigen Befehls SYM-Dateien für mehrere oder alle EXE-Dateien in einem Verzeichnis erzeugen können.

BUILDSYM setzt voraus, daß sich EXEMAP und TMAPSYM entweder im selben Verzeichnis wie BUILDSYM oder auf Ihrem Suchpfad befinden. BUILDSYM schreibt die von ihm erzeugte SYM-Datei in das aktuelle Verzeichnis. Damit WinSpector eine SYM-Datei finden kann, muß sie sich im selben Verzeichnis befinden wie das ausführbare Programm, das die Exception verursacht hat.

BUILDSYM führt folgende Aktionen aus:

- Überprüfung, ob es sich bei den Dateien um Windows-Dateien handelt (falls nicht, werden sie ignoriert)
- Aufruf von EXEMAP zur Erzeugung von MAP-Dateien
- Prüfung, ob MAP-Dateien erzeugt worden sind
- Aufruf von TMAPSYM mit Übergabe der Namen der neuen MAP-Dateien, so daß TMAPSYM SYM-Dateien erzeugen kann
- Löschen der MAP-Dateien, da sie nicht mehr benötigt werden

Geben Sie `BUILDSYM DateiName.EXE` ein, um eine SYM-Datei aus einer EXE-Datei zu erzeugen.

Es kann praktisch sein, wenn Sie DOS-Jokerzeichen bei der Angabe des Dateinamens verwenden. Beispiel: Geben Sie `BUILDSYM *.EXE` ein, um zu allen EXE-Dateien im aktuellen Verzeichnis SYM-Dateien zu erzeugen.

K A P I T E L 9

Arbeiten mit dem Linker: TLINK

Anhang A »Fehlermeldungen« führt die von TLINK und vom in die IDE integrierten Linker ausgegebenen Meldungen auf.

TLINK und TLINK32 sind Dienstprogramme, die von der Kommandozeile aus aufgerufen werden; sie dienen dazu, Objektmodule (OBJ-Dateien) und Bibliotheksmodule (LIB-Dateien) zu ausführbaren Dateien zu verbinden. In die IDE sind Versionen dieser Linker integriert. Da der Compiler automatisch den Linker aufruft, müssen Sie den Befehl TLINK nicht verwenden, *vorausgesetzt*, daß Sie den Link-Vorgang beim Compilieren nicht unterdrückt haben (siehe die Compiler-Option **-c**). Soweit nicht anders angegeben, gelten die Befehle und Optionen von TLINK auch für TLINK32.

Grundsätzliches zu TLINK

Bei den Optionen von TLINK und TLINK32 ist die Groß- u. Kleinschreibung zu beachten.

TLINK benutzt eine Konfigurationsdatei, genannt TLINK.CFG, eine Response-Datei (optional) sowie Kommandozeilenparameter. Mit den darin enthaltenen Informationen linkt TLINK Objektmodule, Bibliotheken und Ressourcen zu einer ausführbaren Datei (.EXE oder .DLL) zusammen. Der IDE-Linker arbeitet mit den bei *Optionen/Projekt* unter Teil *Linker* eingestellten Optionen. Die Syntax zum Aufruf von TLINK lautet:

```
TLINK [@respfile] [optionen] startupdatei meineobjs, exename, [map_datei]
[meinelibs] runtimelib [import], [def_datei], [res_dateien]
```

Dabei gilt:

- *optionen* steht für die Optionen, die steuern, wie TLINK arbeitet. Beispielsweise wird durch Optionen bestimmt, ob eine EXE- oder eine DLL-Datei erzeugt wird. Den Optionen von TLINK muß entweder ein Schrägstrich (**/**) oder ein Minuszeichen (**-**) vorangestellt werden. Um eine Standardoption zu deaktivieren, setzen Sie ein Minuszeichen dahinter (also zum Beispiel **-P-**). Tabelle 9.3 zeigt alle Optionen von TLINK.

Die Tabellen 9.1 und 9.2 geben an, welche Module benutzt werden müssen.

- *startupdatei* bezeichnet ein Initialisierungsmodul von Borland, das für ausführbare Programme oder DLLs bestimmt ist und die Reihenfolge der verschiedenen Segmente des Programms festlegt. Das Initialisierungsmodul muß an erster Stelle der Aufzählung der Objektdateien stehen. Steht es nicht an erster Stelle, so werden die Programmsegmente möglicherweise nicht korrekt im Speicher plaziert, was zu diversen ärgerlichen Programmfehlern führen kann. Wenn Sie nicht das passende Initialisierungsmodul hinzulinken, erhalten Sie gewöhnlich eine lange Liste von Fehlermeldungen, die Ihnen sagen, daß bestimmte Namensreferenzen nicht aufgelöst wurden oder daß kein Stack angelegt wurde.
- *meineobjs* bezeichnet die OBJ-Dateien, die Sie zusammenlinken wollen. Geben Sie den Pfad an, falls die Dateien nicht im aktuellen Verzeichnis liegen.
- *exename* ist der Name, den Sie der ausführbaren Datei (.EXE oder .DLL) geben wollen. Wenn Sie hier keinen Namen angeben, bildet TLINK selbst einen, indem er die Endung .DLL oder .EXE an den Namen der ersten Objektdatei in der Aufzählung anhängt. Geben Sie auf jeden Fall in der Kommandozeile explizit einen Namen für die ausführbare Datei an. Andernfalls kann es Ihnen passieren, daß Ihr Programm beispielsweise den Namen C02.EXE o.ä. bekommt, was Sie vermutlich nicht wollen.
- *mapdatei* (optional) ist der Name, den Sie der Map-Datei geben wollen. Wenn Sie hierfür keinen Namen angeben, so legt ihn TLINK selbst fest und benutzt dafür den Namen von *exefil*e, aber mit der Erweiterung .MAP.
- *meinelibs* bezeichnet die Bibliotheksdateien, die Sie mit hinzulinken wollen. Falls die Dateien nicht im aktuellen Verzeichnis oder Suchpfad (siehe Option **/L**) liegen, müssen Sie hier auch die Pfade angeben.
- *runtimelib* bezeichnet die Laufzeitbibliothek von Borland. Wird hier nichts angegeben, so werden keine Bibliotheken hinzugelinkt.
- *importlib* bezeichnet die Importbibliothek von Windows, die den Zugriff auf die Windows-API-Funktionen von Microsoft ermöglicht.
- *defdatei* bezeichnet die Moduldefinitionsdatei für ausführbare Windows-Programme. Wenn Sie keine DEF-Datei angeben, erzeugt TLINK ein Anwendungsprogramm gemäß den Standardeinstellungen.
- *resdateien* ist eine Liste von RES-Dateinamen, die dem ausführbaren Programm hinzugefügt werden sollen.

TLINK geht bei Dateinamen ohne Erweiterung von folgenden Namenserweiterungen aus oder hängt sie an:

- .OBJ für Objektdateien
- .LIB für Bibliotheksdateien
- .EXE für ausführbare Dateien
- .DEF für Moduldefinitionsdateien
- .DLL für dynamische Linkbibliotheken
- .RES für Ressourcen-Dateien
- .MAP für Map-Dateien

TLINK.CFG

Die IDE arbeitet mit den unter *Optionen/Projekt* und *Optionen/Schablonen* eingegebenen Linkeroptionen. Für weitere Informationen über die Vorgabe von Optionen für ein Projekt siehe Kapitel 2.

TLINK arbeitet mit einer Konfigurationsdatei, TLINK.CFG (oder TLINK32.CFG) genannt, in der Optionen stehen, die Sie normalerweise in der Kommandozeile eingeben müßten. (Hinweis: In TLINK.CFG können nur Optionen, keine Dateinamen stehen.) Eine Konfigurationsdatei speichert Optionen, die Sie häufig benutzen, so daß Sie sie nicht jedesmal neu in der Kommandozeile eingeben müssen.

TLINK sucht nach TLINK.CFG zuerst im aktuellen Verzeichnis und danach in dem Verzeichnis, aus dem heraus TLINK gestartet wurde.

Die folgende Datei TLINK.CFG weist TLINK an, nach Bibliotheken zuerst im Verzeichnis C:\BC4\LIB und danach im Verzeichnis C:\WINAPPS\LIB zu suchen. In die erzeugten ablauffähigen Dateien sollen Debug-Informationen aufgenommen werden, es soll eine detaillierte Segmenttabelle erzeugt werden, und es soll ein ausführbares Windows-Programm (.EXE, nicht .DLL) erzeugt werden.

TLINK	TLINK32
`/Lc:\bc4\lib;c:\winapps\lib`	`/Lc:\bc4\lib;c:\winapps\lib`
`/v /s`	`/v /s`
`/Twe`	`/Tpe`

Response-Dateien

Die Kommandozeilencompiler arbeiten auch mit Response-Dateien.

Response-Dateien sind ASCII-Dateien, die die Optionen und Dateinamen für TLINK (und TLINK32) enthalten, die Sie normalerweise in die Kommandozeile eintippen müßten.

Response-Dateien erlauben es Ihnen, mit einer längeren Kommandozeile zu arbeiten, als es die meisten Betriebssysteme zulassen. Response-Dateien können dieselben Informationen enthalten wie Konfigurationsdateien (Kommandozeilenoptionen), zusätzlich aber auch noch Dateinamen.

Im Gegensatz zur Kommandozeile darf eine Response-Datei mehrere Textzeilen umfassen. Um diese Möglichkeit zu nutzen, beenden Sie jede Zeile mit einem Pluszeichen (+). Achtung: Steht am Ende eine Option, bei der ein Pluszeichen zum Aktivieren dient (wie etwa bei **/v+**), so wird das Pluszeichen nicht als Fortsetzungszeichen für die Zeile interpretiert. (Verwenden Sie dann **/v++** zum Fortsetzen der Zeile.)

Wenn Sie unterschiedliche Elemente der Kommandozeile dadurch gruppieren (zum Beispiel OBJ-Dateien und LIB-Dateien), daß Sie sie in einer Response-Datei in getrennte Zeilen schreiben, so müssen Sie das Komma, das zur Trennung in der Kommandozeile dient, weglassen. In der folgenden Beispieldatei

```
/c c0ws+
   myprog,myexe
   mymap
   mylib cws
```

sind die Kommas weggelassen, die in der gleichwertigen Kommandozeile

```
TLINK /c c0ws myprog,myexe,mymap,mylib cws
```

stehen müssen.

Um mit einer Response-Datei zu arbeiten, tun Sie folgendes:

1. Schreiben Sie die Kommandozeilenoptionen und Dateinamen in eine ASCII-Textdatei, und speichern Sie diese ab.
2. Geben Sie den Befehl **TLINK @[path]RESFILE.RSP** ein, wobei **RESFILE.RSP** der Name der Response-Datei ist.

Sie können auch mehr als eine Response-Datei angeben, und zwar wie folgt:

```
tlink /c @listobjs,myexe,mymap,@listlibs
```

Wenn Sie eine Response-Datei zusätzlich zu Optionen in der Kommandozeile benutzen, so haben die Kommandozeilenoptionen Vorrang vor allen Optionen in der Response-Datei. Wenn also beispielsweise in einer Response-Datei **-v** steht, Sie aber **-v-** in die Kommandozeile schreiben, so arbeitet TLINK mit der Kommandozeilenoption **-v-**.

TLINK mit BCC.EXE benutzen

Sie können an TLINK Optionen und Dateinamen über die Kommandozeilencompiler (BCC.EXE und BCC32.EXE) übergeben, indem Sie die Dateinamen in der Kommandozeile mit expliziten Namenserweiterungen eingeben. In dem Beispiel

BCC startet TLINK stets mit der Option **/c** (Groß-/Kleinschreibung berücksichtigen).

```
BCC mainfile.obj sub1.obj mylib.lib
```

werden MAINFILE.OBJ, SUB1.OBJ und MYLIB.OBJ zu der ausführbaren Datei MAINFILE.EXE zusammengelinkt.

BCC ruft TLINK mit den Dateien C0WS.OBJ, CWS.LIB und IMPORT.LIB (Initialisierungsmodul, Laufzeitbibliothek und Importbibliothek) auf. BCC32 ruft TLINK32 in der Voreinstellung mit den Dateien C0W32.OBJ, CW32.LIB und IMPORT32.LIB auf.

Bibliotheken linken

Sie müssen in jedem Fall die Laufzeitbibliothek von Borland C++ hinzulinken; sie enthält die Standard-Bibliotheksfunktionen von C bzw. C++ für den Typ Ihrer speziellen Anwendung. Außerdem müssen Sie auch die passende Importbibliothek (IMPORT.LIB für 16-Bit-Windows-Anwendungen, IMPORT32.LIB für Textbildschirmanwendungen oder IMPORTW32.LIB für 32-Bit-Windows-Anwendungen) hinzunehmen.

Tabelle 9.1 beschreibt die von Borland gelieferten 16-Bit-Bibliotheken für Windows 3.x und OBJ-Dateien. Das *Borland C++ Referenzhandbuch* enthält

eine vollständige Liste der Windows-Bibliotheken, das *DOS-Referenzhandbuch* eine vollständige Liste der DOS-Bibliotheken und Startup-Dateien.

Tabelle 9.1: Bibliotheken und Startup-Dateien von Borland

Bibliotheken und OBJs	Beschreibung
C*n*.LIB	Laufzeitbibliothek für DOS-Anwendungen; dabei kann *n* S, C, M, L oder H sein, was für eines der Speichermodelle Small, Compact, Medium, Large oder Huge steht.
CW*n*.LIB	Laufzeitbibliothek für Windows-3.x-Anwendungen; dabei kann *n* S, C, M oder L sein, was für eines der Speichermodelle Small, Compact, Medium oder Large steht.
CRTLDLL.LIB	Laufzeitbibliothek für Windows-3.x-Anwendungen, zum Linken als DLL.
IMPORT.LIB	Importbibliothek für Windows-3.x-API-Funktionen.
C0*n*.OBJ	Startup-Modul für DOS-Anwendungen; dabei kann *n* T, S, C, M , L oder H sein, was für eines der Speichermodelle Tiny, Small, Compact, Medium, Large oder Huge steht.
C0W*n*.OBJ	Startup-Modul für Windows-3.x-Anwendungen; dabei kann *n* S, C, M oder L sein, was für eines der Speichermodelle Small, Compact, Medium oder Large steht.
C0D*n*.OBJ	Startup-Modul für Windows-3.x-DLL-Module; dabei kann *n* S, M oder L sein, was für eines der Speichermodelle Small, Medium oder Large steht.
MATHWS.LIB	Wenn Ihr Programm Gleitkomma-Operationen enthält, müssen Sie eine mathematische Bibliothek hinzulinken. MATHWS.LIB ist eine solche Bibliothek für die Speichermodelle Tiny und Small.
MATHWC.LIB	Mathematische Bibliothek für das Speichermodell Compact.
MATHWM.LIB	Mathematische Bibliothek für das Speichermodell Medium.
MATHWL.LIB	Mathematische Bibliothek für das Speichermodell Large.

Tabelle 9.2 beschreibt kurz die von Borland gelieferten 32-Bit-Bibliotheken und OBJ-Dateien, wie sie von TLINK32 verwendet werden. Das *Borland C++ Referenzhandbuch* enthält eine vollständige Liste der Bibliotheken.

Tabelle 9.2: 32-Bit-Bibliotheken und -Startup-Dateien von Borland

Bibliotheken und OBJs	Beschreibung
CW32.LIB	Laufzeitbibliothek für Win32-Anwendungen
IMPORT32.LIB	Importbibliothek für Textbildschirm- und Win32-Anwendungen.
C0X32.OBJ	Startup-Modul für Textbildschirmanwendungen
C0W32.OBJ	Startup-Modul für Win32-Anwendungen
C0D32.OBJ	Startup-Modul für 32-Bit-DLL-Module3

Optionen für TLINK

Soweit nicht anders angegeben, gelten die Optionen sowohl für TLINK als auch für TLINK32. Bei den Optionen wird zwischen Groß- und Kleinschreibung unterschieden, ihnen muß ein Schrägstrich (**/**) oder ein Minuszeichen (**-**) vorangestellt werden. Um eine Standardoption zu deaktivieren, setzen Sie ein Minuszeichen dahinter (also zum Beispiel **-P-** oder **/P-**). Optionen dürfen überall in der Kommandozeile stehen, Leerzeichen zwischen ihnen sind nicht erforderlich (**/m/f/c** ist gleichbedeutend mit **/m /f /c**), allerdings müssen Optionen und Dateinamen voneinander durch ein Leerzeichen getrennt sein.

Tabelle 9.3 zeigt die Kommandozeilenoptionen für TLINK und die dazu äquivalenten Optionen des IDE-Linkers (nicht alle Kommandozeilenoptionen haben ein solches Äquivalent). Die voreingestellten Kommandozeilenoptionen sind durch das Symbol $ markiert. Im Anschluß an die Tabelle folgt eine detailliertere Erläuterung der Optionen einschließlich der Bezeichnungen der IDE-Optionen.

Tabelle 9.3: Optionen für TLINK

Voreinstellung	Option	Option für IDE-Linker	Für	Beschreibung
	/3	Linker/16-Bit-Linker/Enable 32-Bit-Processing	16 Bit	Akzeptiert und linkt 32-Bit-Programme, erzeugt von TASM oder dazu kompatiblen Assembler.
	/ax	TLINK Target Attributes/Target Model	32 Bit	Gibt den Typ des Anwendungsprogramms an, dabei steht **/aa** für Windows-Anwendungen, **/ap** für Textbildschirmanwendungen.
	/A:*dd*	16-bit Linker/Segmentausrichtung	16/32 Bit	Legt die Seitenausrichtung in der EXE-Datei fest.
	/B:*xxxxxx*	32-bit Linker/Image-Basisadresse	32 Bit	Legt die Basisadresse (hexadezimal) für das Speicherabbild fest.
	/c	Allgemein/Groß-/Kleinschreibung beim Linken beachten	16/32 Bit	Unterscheidung von Groß- und Kleinschreibung in Symbolen.
	/C	Allgemeines/Groß\Kleinschr. bei Export/Import beachten	16 Bit	Unterscheidung von Groß- und Kleinschreibung im EXPORTS- und IMPORTS-Abschnitt von Moduldefinitionsdateien.
	/d	Warnungen/Warnung bei doppelten Symbolen in LIB-Datei	16 Bit	Warnung, wenn gleichnamige Symbole mehrfach in Bibliotheken auftreten.
	/E	16-bit Linker/Erweitertes Wörterbuch benutzen	16 Bit	Aktiviert die Verarbeitung von erweiterten Wörterbüchern in Bibliotheken.

	/E*nn*	32-bit Linker/Maximal erlaubte Linker-Fehler	32 Bit	Maximal tolerierte Fehleranzahl vor Abbruch.
■	**/e**	16-bit Linker/Erweitertes Wörterbuch benutzen	16 Bit	Ignoriert erweiterte Wörterbücher (deaktiviert)in Bibliotheken (Gegenteil der Option **/E**).
	/f	16-bit Linker/Optimierung von Far- zu Near-Aufrufen verhindern	16 Bit	Unterdrückt die Optimierung von Far-Aufrufen zu Near-Daten.
	/G*x*		16/32 Bit	Linker-Extras, dabei steht *x* für n, r oder m.
	/Gn	16-bit Linker/Tabelle nicht-residenter Namen entfernen		16 BitTabelle der nichtresidenten Namen verwerfen.
	/Gr	16-bit Linker/Residente Namen übertragen	16 Bit	Residente Namen in die Tabelle der nichtresidenten Namen übertragen.
	/Gm	Map-Datei/Abgekürzte Namen in Map-Datei schreiben	16/32 Bit	Abgekürzte Namen in die Map-Datei schreiben.
	/i	16-bit Linker/Segmente initialisieren	16 Bit	Alle Segmente initialisieren.
	/l	Map-Datei/Zeilennummern mit aufnehmen	16 Bit	Quelltextzeilennummern einfügen (kleines L!).
	/L	Verzeichnisse/Bibliothek (nicht unter *Linker* in IDE)	16/32 Bit	Legt die Bibliothekssuchpfade fest.
	/m	Map-Datei/Publics	16/32 Bit	Map-Datei mit Public-Symbolen erzeugen.
	/n	Allgemein/Standardbibliotheken	16 Bit	Keine Standardbibliotheken benutzen.
	/o	Overlay-Modul (Dialogfenster *Node attributes*)	16 Bit	Overlay-Module oder Bibliotheken.
■	**/P**	Allgemein/Code-Segmente komprimieren	16 Bit	Codesegmente komprimieren.
	/Rk	Ressourcen/Fastload-Bereich packen (nicht unter *Linker*)	16 Bit	Optionen an RLINK.EXE übergeben.
	/Rv		32 Bit	Text-Ressourcen hinzufügen.
	/Re*xxxx*		32 Bit	Ausführbares Programm in xxxx umbenennen.
	/S:*xxxxxx*	32-bit Linker/Stack-Größe	32 Bit	Spezifiziert die Größe des Stack (hexadezimal).
	/s	Map-Datei/Detailliert	16 Bit	Detaillierte Liste der Segmente erzeugen.
	/t		16 Bit	COM-Datei (für DOS) im Speichermodell Tiny erzeugen.
■	**/Td***x*	TargetExpert Platform	16 Bit	Spezifiziert das erzeugte Anwendungsprogramm, dabei steht **/Tdc** für eine COM-Datei (DOS), **/Tde** für eine EXE-Datei (DOS).
	/Tp*x*	TargetExpert Platform	32 Bit	Spezifiziert das erzeugte Anwendungsprogramm, dabei steht

			/Tpe für eine 32-Bit-EXE-Datei, **/Tpd** für eine 32-Bit-DLL.
/Tw*x*	TargetExpert Target Type	16 Bit	Spezifiziert die erzeugte Windows-3.x-Anwendung, dabei steht **/Twe** für eine Windows-EXE-Datei **/Twd** für eine Windows-DLL.
/v	Allgemein/Debug-Informationen mit aufnehmen	16/32 Bit	Vollständige symbolische Debug-Information einfügen.
/w*xxx*	Warnungen	32 Bit	Warnmeldungen freigeben oder unterdrücken.
/x	Map-Datei/Aus	16/32 Bit	Keine Map-Datei erzeugen.
/ye		16 Bit	Expanded Memory zum Auslagern verwenden.
/yx			Konfiguriert die Verwendung des Extended Memory durch TLINK.

/3 (32-bit code = 32-Bit-Code) ermöglicht es Ihnen, 32-Bit-OBJ-Module (DOS), die von TASM oder einem kompatiblen Assembler erzeugt wurden, zu linken. Bei Benutzung dieser Option steigt der Speicherbedarf für TLINK, und die Linkzeit erhöht sich.

/a (application type) Typ des Anwendungsprogramms) erlaubt es Ihnen, den Typ der EXE-Datei zu bestimmen:

- **/aa** erzeugt Windows-Anwendungsprogramme
- **/ap** erzeugt Textbildschirmanwendungen, die in einem Fenster ablaufen können.

/A:dd (align pages = Seitenausrichtung) spezifiziert die Seitenausrichtung von Befehlen und Daten in der ausführbaren Datei. Dabei muß *dd* eine Dezimalzahl sein, die als Exponent von 2 interpretiert wird. Die Vorgabe **/A:12** bewirkt beispielsweise, daß die Sektionen des Speicherabbildes nach 4096-Byte-Grenzen ausgerichtet werden. Das Betriebssystem sucht beim Laden auf der Grundlage dieses Wertes nach freien Seiten. Die Voreinstellung ist **/A:9**, damit werden die Seiten innerhalb der ausführbaren Datei auf 512-Byte-Grenzen ausgerichtet.

/B:xxxxxx (base address = Basisadresse) legt die Basisadresse des Speicherabbildes eines Anwendungsprogramms fest. Wenn diese Option gewählt wird, werden interne Fixups aus dem Speicherabbild entfernt, und die dafür geforderte Ladeadresse der ersten OBJ-Datei wird auf den in dieser Option angegebenen Wert gesetzt. Alle nachfolgenden OBJ-Dateien werden auf 64kB-Grenzen im linearen Adreßraum ausgerichtet. Diese Option führt zu einer kürzeren Dateilänge auf der Platte und verbessert die Lade- und Laufzeiten, weil das Betriebssystem keine internen Fixups mehr einsetzen muß. Da Windows NT alle EXE-Speicherabbilder

auf 64kB-Grenzen lädt, ist es empfehlenswert, alle EXE-Dateien mit der Option **/B:0x10000** zu linken.

/c (case sensitivity = Groß-/Kleinschreibung beachten) bewirkt, daß bei öffentlichen und externen Symbolen zwischen Groß- und Kleinschreibung unterschieden wird.

/C (case sensitivity = Groß-/Kleinschreibung beachten) bewirkt, daß in den EXPORTS- und IMPORTS-Abschnitten von Moduldefinitionsdateien zwischen Groß- und Kleinschreibung unterschieden wird.

/d (duplicate symbols = Doppelte Symbole) bewirkt, daß Sie eine Warnung erhalten, wenn dasselbe Symbol in mehr als einer Bibliotheksdatei vorhanden ist. Wenn das Symbol in das Programm aufgenommen werden muß, so wählt TLINK dazu dasjenige, das als erstes angetroffen wird, wenn die in der Kommandozeile (oder in einer Response-Datei) angegebenen Dateien der Reihe nach durchgegangen werden. Diese Option erzeugt auch Warnungen für den Fall, daß gleichnamige Symbole sowohl in einer OBJ-Datei als auch in einer LIB-Datei gefunden werden (TLINK wählt dann das erste und ignoriert die anderen).

/Enn (maximum errors = Maximale Fehlerzahl) spezifiziert die maximale Anzahl von Fehlern, die auftreten dürfen, bevor der Linker abbricht. Die Voreinstellung **/E0** bedeutet, daß eine unbegrenzte Zahl (also so viele wie möglich) gemeldet werden.

/E (extended dictionaries = Erweiterte Wörterbücher) bewirkt, daß erweiterte Wörterbücher verarbeitet werden. Die Bibliotheksdateien von Borland C++ enthalten ein erweitertes Wörterbuch mit Informationen, die TLINK beim Linken solcher Bibliotheken zum schnelleren Linken bei geringerem Speicherbedarf nutzt. Verwenden Sie die Option **/E** von TLIB (siehe den Abschnitt TLIB), um ein erweitertes Wörterbuch auch zu anderen Bibliotheken hinzuzufügen. Vermeiden Sie es, die Option **/E** mit Programmen zu benutzen, die mit Bibliotheken gelinkt sind, die ohne ein solches erweitertes Wörterbuch aufgebaut wurden (zum Beispiel Bibliotheken, die ohne Quellprogramme von Fremdlieferanten geliefert wurden). Damit erweiterte Wörterbücher benutzt werden können, müssen sie in allen gelinkten Bibliotheken vorhanden sein.

/e (ignore extended dictionaries = Erweiterte Wörterbücher ignorieren) bewirkt, daß erweiterte Wörterbücher ignoriert werden. Diese Option stellt das Gegenteil von **/E** dar, sie ist voreingestellt.

/f (inhibit far optimizations = Verhindern von Far-Optimierungen) verhindert die Optimierung von Far-Aufrufen bei Near-Daten.

/Gx (Goodies = Besonders Nützliches) ist eine Spezialoption mit folgenden Funktionen:

n = Tabelle der nichtresidenten Namen verwerfen.
r = Residente Namen in die Tabelle der nichtresidenten Namen übertragen.
m = (TLINK32) Verkürzte Namen in Map-Datei aufnehmen; dies ermöglicht es Ihnen, leichter zu erkennen, wie die Namen modifiziert werden.

/i (uninitialized trailing segments = Nichtinitialisierte Endsegmente) bewirkt, daß auch uninitialisierte Endsegmente in die ausführbare Datei aufgenommen werden, selbst wenn sie keine Daten enthalten.

/l (line numbers = Zeilennummern) bewirkt die Erzeugung eines Abschnitts in der Map-Datei, der die Zeilennummern des Quelltextes enthält. Die gelinkten OBJ-Dateien müssen dafür Debug-Informationen enthalten (compiliert mit **-y** oder **-v**). Wenn Sie **/x** angegeben haben, damit keine Map-Datei erzeugt wird, hat die Option **/l** (kleines L) keine Wirkung.

/L (library search paths = Suchpfade für Bibliotheken) dient dazu, eine Liste mit den Verzeichnissen anzugeben, in denen TLINK nach Bibliotheken suchen soll, falls Sie keinen expliziten Suchpfad für eine Bibliothek oder das C- bzw. C++ Initialisierungsmodul angegeben haben. TLINK sucht zuerst im aktuellen Verzeichnis (in dem Sie TLINK eingetippt haben). Bei der Kommandozeile

```
TLINK /Lc:\BC4\lib;c:\mylibs splash logo,,,utils .\logolib
```

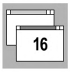
sucht TLINK nach UTILS.LIB zuerst im aktuellen Verzeichnis, danach in C:\BC4\LIB und schließlich in C:\MYLIBS. Da für LOGOLIB das aktuelle Verzeichnis explizit angegeben ist, durchsucht TLINK die mit der Option **/L** festgelegten Verzeichnisse nicht, um LOGOLIB.LIB zu finden.

/m (detailed map file = Ausführliche Map-Datei) bewirkt, daß TLINK eine ausführlichere Map-Datei anlegt als normalerweise. Dabei wird eine zusätzliche sortierte Liste der öffentlichen Symbole in die Map-Datei geschrieben. Eine solche Map-Datei ist beim Debuggen sehr nützlich. Viele Debugger bieten Ihnen über diese Liste öffentlicher Symbole während des Debuggens Zugriff auf symbolische Adressen. Wenn Sie keine der Map-Datei-Optionen (**/m**, **/s** oder **/x**) angeben, wird die unter *Map-Datei/Segmente* eingestellte Option benutzt. Siehe hierzu auch die Option **/s**.

/M (map mangled = Modifizierte Map-Datei) bewirkt, daß verkürzte Namen der öffentlichen Symbole in die Map-Datei geschrieben werden.

/n (ignore default libraries = Standardbibliotheken ignorieren) weist den Linker an, die von einigen Compilern spezifizierten Standardbibliotheken zu ignorieren. Verwenden Sie diese Option dann, wenn Sie Module linken, die in einer anderen Sprache geschrieben sind.

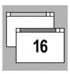
/o (overlays = Overlays) bewirkt, daß Programmteile in allen den Modulen und Bibliotheken überlagert werden, die in der Kommandozeile hinter der Option aufgeführt sind. Diese Option ist nur für DOS-Anwendungsprogramme wirksam. Geben Sie in der Kommandozeile **/o-** ein, um ohne Overlays zu arbeiten. Wenn Sie hinter dieser Option einen Klassennamen angeben, werden alle Segmente dieser Klasse überlagert; Sie können dies für mehrere Klassen tun. Wenn Sie hinter dieser Option gar keinen Namen angeben, so werden alle Segmente aus Klassen, deren Namen auf CODE enden, überlagert. Diese Option verwendet als Voreinstellung die Interruptnummer 3FH. Die Verwendung eines anderen Interrupts erreichen Sie durch **/o#***xx*, wobei *xx* eine zweistellige Hexadezimalzahl ist.



/P (pack code segments = Code-Segmente komprimieren) bewirkt, daß der Linker so viele Code-Segmente wie möglich in ein physikalisches Segment »zusammenpackt«, und zwar bis zur dafür vorgesehenen Packgröße. Wenn nötig, beginnt TLINK ein neues Segment. Die voreingestellte Grenze für das Zusammenfassen von Code-Segmenten liegt bei 8192 Bytes (8 KByte). Zur Änderung dieser Größe geben Sie **/P=n** an, wobei *n* die Anzahl der Bytes von 1 bis 65.536 ist. Im erweiterten 386er-Modus werden Sie wahrscheinlich die Größe auf ein Vielfaches von 4 KByte einstellen wollen, da das Betriebssystem den Speicher in eben diesen Stufen verwaltet.

Obwohl die optimale Segmentgröße im erweiterten 386er-Modus 4 KByte beträgt, ist 8 KByte die Voreinstellung für die Zusammenfassung der Code-Segmente. Da die Größe typischer Code-Segmente zwischen 4 und 8 KByte liegt, ist eine Größe von 8 KByte für das Zusammenfassen wahrscheinlich effektiver.

Da für jedes verwendete Segment ein gewisser Verwaltungsaufwand für das System entsteht, erhöht sich durch die Segmentzusammenfassung wegen der Reduzierung der Anzahl zu verwaltender Segmente in der Regel die Ausführungsgeschwindigkeit. Das Ausschalten der Option mit **/P-** ist praktisch, wenn die Option in der Konfigurationsdatei eingeschaltet wurde, und Sie sie für einen ganz bestimmten Linkvorgang deaktivieren wollen.



/s (detailed segment map = Ausführliche Segmentliste) bewirkt, daß eine Map-Datei mit denselben Eigenschaften wie bei der Option **/m** erzeugt wird; zusätzlich wird aber eine detaillierte Segmentliste erzeugt. Wenn Sie keine der Map-Datei-Optionen (**/m**, **/s** oder **/x**) angeben, wird die unter *Map-Datei/Segmente* eingestellte Option benutzt. Die Segmentliste enthält für jedes Segment in jedem Modul die Adresse, die Länge in Bytes, die Klasse, den Segmentnamen, die Gruppe, das Modul sowie ACBP-Informationen. Wenn ein Segment zu mehreren Modulen gehört, erscheint für jedes Modul eine eigene Zeile. Mit Ausnahme der ACBP-Informationen sind die Angaben in der detaillierten Segmentliste selbsterklärend.

Im ACBP-Feld sind die Attribute A (Ausrichtung), C (Kombination) und B (Größe) nach Vorgaben von Intel in einer Anzahl von 4-Bit-Feldern ko-

diert. TLINK benutzt nur drei dieser Attribute, nämlich A, C und B. Der ACBP-Wert erscheint in der Liste hexadezimal kodiert. Die folgenden Werte der Felder müssen durch ein logisches ODER zusammengefaßt werden, um den ACBP-Wert der Liste zu erhalten:

Feld	Wert	Beschreibung
Feld A (Ausrichtung)	00	Absolutes Segment
	20	An Bytegrenze ausgerichtetes Segment
	40	An Wortgrenze ausgerichtetes Segment
	60	An Paragraphen-Grenze (16 Bytes) ausgerichtetes Segment
	80	An einer Speicherseite ausgerichtetes Segment
	A0	Ein unbenannter, absoluter Speicherbereich
Feld C (Kombination)	00	Kann nicht kombiniert werden
	08	Kombiniert öffentliche Variablen
Feld B (Größe)	00	Segment kleiner als 64 kB
	02	Segment exakt 64 kB groß

Bei Verwendung der Option **/s** werden öffentliche Symbole, die nicht referenziert werden, als *idle* (»unbenutzt«) gekennzeichnet. Ein unbenutztes Symbol ist in einem Modul als öffentlich definiert, aber kein anderes mitgelinktes Modul nimmt per EXTDEF-Datensatz darauf Bezug. Der folgende Auszug des Abschnitts für öffentliche Symbole einer Map-Datei zeigt zum Beispiel, daß die Symbole *Symbol1* und *Symbol3* in dem zusammengelinkten Speicherabbild nicht referenziert werden:

```
0002:00000874   Idle   Symbol1
0002:00000CE4          Symbol2
0002:000000E7   Idle   Symbol3
```

/S:xxxxxx (stack size = Stack-Größe) gibt die Größe des Anwendungs-Stacks in hexadezimaler Form an. Dabei ist *xxxxxx* ein String aus Hexadezimalziffern. Die Vorgabe der Stack-Größe durch **/S** hat Vorrang vor jeder Angabe einer Stack-Größe in einer Moduldefinitionsdatei.

/t (tiny model DOS .COM file = COM-Datei für DOS im Modell Tiny) erzeugt eine COM-Datei (DOS) im Speichermodell Tiny. Diese Option wirkt wie **/Tdc**, mit dem Unterschied, daß Sie **/t** zusammen mit BCC.EXE verwenden können. Eine COM-Datei für DOS kann nicht größer als 64KB sein, sie kann auch keine segmentrelativen Fixups enthalten, und es kann für sie kein Stacksegment definiert sein. Ihre Startadresse muß 0:100H sein. Wenn Sie die Namenserweiterung der Datei ändern (zum Beispiel in BIN), darf die Startadresse wahlweise 0:0 oder 0:100H sein. Der Linker kann für COM-Dateien keine Debug-Informationen erzeugen; für die Fehlersuche müssen Sie also das Programm zunächst zum Testen als EXE-Datei erzeugen und später neu als COM-Datei compilieren und linken.

/Tdx (DOS target = DOS-Zieldatei) erzeugt ein DOS-Programm, und zwar eine EXE-Datei (bei **/Tde**) oder eine COM-Datei (bei **/Tdc**).

/Tpx (protected target = Zieldatei für Protected Modus) bewirkt, daß eine EXE-Datei für den Protected Mode (**/Tpe**) oder eine DLL-Datei (**/Tpd**) erzeugt wird.

/Twx (target type = Zieldateityp) weist TLINK an, eine Windows-EXE-Datei (**/Twe**) oder eine DLL-Datei (**/Twd**)zu erzeugen. Diese Option ist dann nicht erforderlich, wenn Sie eine Moduldefinitionsdatei mit einer EXETYPE Windows-Anweisung benutzen. Enthält die Moduldefinitionsdatei nämlich eine NAME-Anweisung, so erzeugt TLINK ein Anwendungsprogramm (.EXE), enthält die Moduldefinitionsdatei dagegen eine LIBRARY-Anweisung, so erzeugt TLINK eine DLL.

/v (debugging information = Debug-Information) weist den Linker an, Debug-Informationen in die ausführbare Datei aufzunehmen. Steht diese Option irgendwo in der Kommandozeile, werden alle in den Objektmodulen enthaltenen Debug-Informationen in die ausführbare Datei aufgenommen. Verwenden Sie die Optionen **/v+** und **/v-**, um das Aufnehmen der Debug-Informationen für jedes Modul getrennt zu aktivieren oder zu deaktivieren (jedoch nicht in einer Zeile, in der schon **/v** steht). So bewirkt beispielsweise der Befehl

```
TLINK mod1 /v+ mod2 mod3 /v- mod4
```

die Aufnahme von Debug-Informationen für die Module *mod2* und *mod3*, nicht aber für *mod1* und *mod4*.

/wxxx (warning control = Freigabe/Unterdrückung von Warnungen) erlaubt es Ihnen, die Warnmeldungen von TLINK freizugeben (**/wxxx**) oder zu unterdrücken (**/w-xxx**). Dabei steht *xxx* für eines der folgenden Kürzel (»Standard« bedeutet dabei, daß TLINK die Warnung ausgibt, ohne daß Sie sie explizit freigeben müssen):

Tabelle 9.4: Warnungen von TLINK32

Standard	/w-Option	IDE-Warnung
■	**bdl**	Using based linking in DLLs (might cause the DLL to malfunction)
	def	No .DEF file; using defaults
	dpl	Warn duplicate symbol in .LIB
■	**dup**	Duplicate symbol (warning for .OBJs)
■	**ent**	No entry point
	imt	Import doesn't match previous definition
	ing	Extern not qualified with __import
■	**srf**	Self-relative fixup overflowed

/x (no map file = Keine Map-Datei) weist TLINK an, keine Map-Datei zu erzeugen. Standardmäßig erzeugt TLINK eine Map-Datei, in der die Segmente des Programms, seine Startadresse sowie alle Fehler- oder War-

nungsmeldungen von TLINK aufgelistet sind. In der Voreinstellung ist die Option *Map-Datei/Segmente*, die kein Gegenstück als Kommandozeilenoption hat, aktiviert.

/ye (expanded memory = Expanded Memory) steuert die Nutzung des Expanded Memory durch TLINK für das Puffern von Ein- und Ausgabedaten. Wenn TLINK (beim Lesen von OBJ-Dateien oder beim Schreiben der EXE-Datei) für seine aktiven Datenstrukturen mehr Speicher benötigt, werden die Puffer entweder freigemacht oder in den Expanded Memory ausgelagert.

Beim Puffern von Eingabedateien bedeutet »Freimachen« einfach das Verwerfen des Eingabepuffers, so daß der freigewordene Platz für andere Datenstrukturen nutzbar wird. Beim Puffern von Ausgabedateien bedeutet »Freimachen«, daß der Pufferinhalt an seine korrekte Position in der ausführbaren Datei geschrieben wird. In beiden Fällen können Sie die Linkgeschwindigkeit durch Auslagern der Puffer in den Expanded Memory wesentlich erhöhen. Standardmäßig wird zum Auslagern der Expanded Memory benutzt, nicht aber der Extended Memory. Ist das Auslagern aktiviert, ohne daß genügend Speicher zum Auslagern vorhanden ist, so erfolgt keine Auslagerung.

Die Option **/y** gibt es in folgenden Formen:

- **/ye** oder **/ye+** schaltet das Auslagern in den Expanded Memory ein (dies ist die Voreinstellung).
- **/ye-** schaltet das Auslagern in den Expanded Memory aus.

/yx (extended memory = Extended Memory) steuert die Nutzung des Extended Memory durch TLINK für das Puffern von Ein- und Ausgabedaten. Standardmäßig benutzt TLINK bis zu 8 MByte des Extended Memory. Sie können die Verwendung des Extended Memory durch TLINK mit einer der folgenden Formen dieser Option beeinflussen:

- **/yx**+ Den gesamten verfügbaren Extended Memory bis zu 8 MB benutzen.
- **/yxn** Nur bis zu *n* KByte des Extended Memory benutzen.

Die Moduldefinitionsdatei

Das Dienstprogramm IMPDEF erzeugt Moduldefinitionsdateien, aus denen das Dienstprogramm IMPLIB dann Importbibliotheken erzeugt.

Dieser Abschnitt beschreibt die Moduldefinitionsdatei und erläutert die darin enthaltenen Anweisungen. Die Moduldefinitionsdatei liefert dem Linker Informationen über den Inhalt und die Systemanforderungen einer Windows-Anwendung. Im einzelnen enthält eine Moduldefinitionsdatei folgendes:

- Name der EXE- oder DLL-Datei
- Festlegung des Typs der Anwendung
- Listen der importierten und exportierten Funktionen
- Beschreibung der Code- und der Datensegmentattribute; für zusätzliche Code- und Datensegmente können Sie Attribute angeben
- Festlegung der Größe des Stacks
- Möglichkeiten zur Einfügung eines DOS-Stub-Programms

Siehe auch das *Borland C++ Programmierhandbuch* für Anwendungshinweise und Beispiele für Moduldefinitionsdateien.

Anweisung CODE

Die Anweisung CODE definiert die Standardattribute der Code-Segmente. Code-Segmente dürfen beliebige Namen haben, müssen aber zu Segmentklassen gehören, deren Namen mit CODE enden (wie zum Beispiel CODE oder MYCODE). Die Syntax von CODE lautet:

TLINK

```
CODE [FIXED | MOVEABLE]
     [DISCARDABLE | NONDISCARDABLE]
     [PRELOAD | LOADONCALL]
```

TLINK32

```
[PRELOAD | LOADONCALL]
[EXECUTEONLY | EXECUTEREAD]
```

- FIXED (Voreinstellung) bedeutet, daß das Segment an einer festen Position im Speicher bleibt; MOVEABLE dagegen besagt, daß das Segment verschoben werden darf.
- PRELOAD bedeutet, daß das Segment geladen wird, wenn das Modul erstmalig geladen wird. Bei LOADONCALL (Voreinstellung) wird das Segment erst dann geladen, wenn Code aus diesem Programm aufgerufen wird.
- DISCARDABLE bedeutet, daß das Segment verworfen werden darf, falls es nicht länger benötigt wird (dadurch wird auch MOVEABLE impliziert). NONDISCARDABLE (Voreinstellung) bedeutet, daß das Segment nicht verworfen werden darf.
- EXECUTEONLY bedeutet, daß das Code-Segment nur ausgeführt werden darf. EXECUTEREAD (Voreinstellung) bedeutet, daß das Segment ausgeführt oder gelesen werden darf.

Anweisung DATA

Die Anweisung DATA definiert Attribute für Datensegmente. Die Syntax lautet:

```
DATA [NONE | SINGLE | MULTIPLE]
          [READONLY | READWRITE]
          [PRELOAD | LOADONCALL]
          [SHARED | NONSHARED]
```

- NONE bedeutet, daß kein Datensegment erzeugt wird. Diese Option ist nur für Bibliotheken verfügbar. SINGLE (Voreinstellung für DLLs) be-

deutet, daß nur ein einziges Datensegment erzeugt wird, auf das alle Prozesse gemeinsam zugreifen. MULTIPLE (Voreinstellung für EXE-Dateien) heißt, daß für jeden Prozeß ein eigenes Datensegment erzeugt wird.

- READONLY bedeutet, daß das Datensegment nur gelesen werden darf. READWRITE (Voreinstellung) bedeutet, daß es gelesen und beschrieben werden darf.
- PRELOAD bedeutet, daß das Segment geladen wird, wenn ein Modul, das das Segment benötigt, erstmalig geladen wird. Bei LOADONCALL (Voreinstellung) wird das Segment erst dann geladen, wenn der erste Zugriff darauf erfolgt (diese Einstellung wird bei 32-Bit-Anwendungen ignoriert).
- SHARED (Voreinstellung für 16-Bit-DLLs) bedeutet, daß alle Prozesse auf dieselbe Kopie des Datensegments zugreifen. NONSHARED (Voreinstellung für Programme und 32-Bit-DLLs) bedeutet, daß für jeden Prozeß, der das Datensegment benutzen muß, eine eigene Kopie des Datensegments geladen wird.

Anweisung DESCRIPTION

Die Anweisung DESCRIPTION (optional) fügt Text in das Anwendungsmodul ein. Diese Anweisung wird normalerweise verwendet, um den Autor, das Datum oder Copyright-Informationen einzufügen. Die Syntax lautet:

```
DESCRIPTION 'Text'
```

Text steht für einen ASCII-String, der von einfachen Anführungszeichen eingeschlossen wird.

Anweisung EXETYPE

Die Anweisung EXETYPE legt die Voreinstellung des Header-Typs der ausführbaren Datei (.EXE) für 16-Bit-Anwendungsprogramme fest. Sie können diese Anweisung zur Abwärtskompatibilität bei 32-Bit-Programmen in der Moduldefinitionsdatei stehen lassen; wenn Sie die Anweisung EXETYPE aber ändern müssen, so lesen Sie dazu auch den Abschnitt über die NAME-Anweisung. Die Syntax für EXETYPE lautet:

```
EXETYPE WINDOWS
```

Anweisung EXPORTS

Die Anweisung EXPORTS definiert Namen und Attribute der zu exportierenden Funktionen. Das Schlüsselwort EXPORTS markiert den Beginn der Definitionen. Danach kann eine beliebige Anzahl von Exportdefinitionen folgen, wobei jede Definition in einer separaten Zeile stehen muß. Die Syntax lautet:

```
EXPORTS
    ExportName [Ordinal] [RESIDENTNAME] [Parameter]
```

- *ExportName* steht für einen ASCII-String, der das zu exportierende Symbol definiert und folgendes Format hat:

  ```
  EinsprungName [=InternerName]
  ```

 InternerName ist der in der Anwendung für den Bezug auf diesen Eintrag verwendete Name. *EinsprungName* ist der in der Einsprungtabelle der ausführbaren Datei aufgeführte Name, er ist extern sichtbar.

- *Ordinal* definiert den Ordinalwert der Funktion und hat folgendes Format:

  ```
  @Ordinal
  ```

 Ordinal steht für einen Integerwert, der den Ordinalwert der Funktion bestimmt.

 Wenn ein Anwendungs- oder DLL-Modul eine aus einer DLL exportierte Funktion aufruft, kann das aufrufende Modul die Funktion entweder über den Namen oder den Ordinalwert ansprechen. Dabei ist ein Aufruf über den Ordinalwert schneller, da für das Auffinden der Funktion kein Stringvergleich erforderlich ist. Auch in bezug auf die Speicherbelegung ist das Exportieren einer Funktion über den Ordinalwert (vom Standpunkt der Linkbibliothek dieser Funktion aus gesehen) sowie das Importieren/Aufrufen (vom aufrufenden Modul aus gesehen) effizienter. Um Speicher zu sparen, exportieren Sie eine Funktion über den Ordinalwert (aus der Sicht der DLL, in der die Funktion liegt), und importieren bzw. rufen Sie eine Funktion über den Ordinalwert (aus der Sicht des aufrufenden Moduls) auf.

 Wenn eine Funktion unter Verwendung ihres Ordinalwertes exportiert wird, wird der Name in der nichtresidenten Namenstabelle abgelegt. Wird eine Funktion unter Verwendung ihres Namens exportiert, steht der Name in der residenten Namenstabelle. Die residente Namenstabelle für ein Modul ist immer dann im Speicher, wenn das Modul geladen ist; die nichtresidente Namenstabelle dagegen nicht.

- RESIDENTNAME legt fest, daß der Funktionsname jederzeit resident sein muß. Das ist nur dann sinnvoll, wenn die Funktion über ihren Ordinalwert exportiert wird (d.h. wenn der Name nicht per Voreinstellung resident ist).
- *Parameter* ist ein optionaler Integerwert, der die Anzahl WORDs angibt, die die Funktion als Argumente erwartet.

Anweisung IMPORTS

Die Anweisung IMPORTS definiert die Namen und Attribute der Funktionen, die aus DLLs importiert werden sollen. Wenn Sie die zu importierenden Linkbibliotheksfunktionen nicht in der IMPORTS-Anweisung auflisten wollen, erreichen Sie dies entweder

- durch Angabe einer Importbibliothek für die DLL (in der TLINK-Kommandozeile) oder
- durch Aufnehmen der Importbibliothek für die DLL (im Projektmanager der IDE).

Wenn Sie 32-Bit-Programme schreiben, müssen Sie **__import** für den Import jeder Funktion, jeder Klasse und aller Daten, die Sie importieren wollen, benutzen. Bei 16-Bit-Programmen müssen Sie **__import** für Klassen benutzen. Siehe das *Borland C++ Programmierhandbuch* für weitere Informationen über **__import**.

Das Schlüsselwort IMPORTS markiert den Beginn der Definitionen. Ihm folgt eine beliebige Anzahl von Importdefinitionen, wobei jede Definition in einer separaten Zeile stehen muß. Die Syntax lautet:

```
IMPORTS
     [InternerName=] ModulName.Einsprung
```

- *InternerName* ist ein ASCII-String, der den eindeutigen Namen festlegt, mit dem die Anwendung die Funktion aufruft.
- *ModulName* definiert einen oder mehrere ASCII-Zeichen (Großbuchstaben), die das ausführbare Modul benennen, in dem die Funktion liegt. Der Modulname muß dem Namen der ausführbaren Datei entsprechen. So hat die Datei SAMPLE.DLL z.B. den Modulnamen SAMPLE.
- *Einsprung* spezifiziert die Funktion, die importiert werden soll. Dabei kann es sich entweder um einen ASCII-String für den Namen der Funktion oder um einen Integerwert für den Ordinalwert der Funktion handeln.

Anweisung LIBRARY

Die Anweisung LIBRARY bestimmt den Namen eines DLL-Moduls. Eine Moduldefinitionsdatei kann entweder eine LIBRARY-Anweisung, die eine DLL bezeichnet, oder eine NAME-Anweisung, die ein Programm bezeichnet, enthalten.

Ein Bibliotheksmodulname muß dem Namen der ausführbaren Datei entsprechen. So hat z.B. die Bibliothek MYLIB.DLL den Modulnamen MYLIB. Die Syntax lautet:

```
LIBRARY Bibliotheksname [INITGLOBAL | INITINSTANCE]
```

- *Bibliotheksname* (optional) ist ein ASCII-String, der den Namen des Bibliotheksmoduls festlegt. Wenn Sie *Bibliotheksname* nicht angeben, verwendet TLINK den Namen der ausführbaren Datei, jedoch ohne die Namenserweiterung. Enthält die Moduldefinitionsdatei weder eine NAME- noch eine LIBRARY-Anweisung, so verhält sich TLINK wie bei einer NAME-Anweisung ohne den Parameter *ModulName*.

- INITGLOBAL bedeutet, daß das Unterprogramm zur Initialisierung der Bibliothek nur dann aufgerufen wird, wenn das Bibliotheksmodul zum ersten Mal geladen wird. INITINSTANCE bedeutet, daß das Unterprogramm zur Initialisierung der Bibliothek jedesmal aufgerufen wird, wenn ein neuer Prozeß die Bibliothek aufruft.

Anweisung NAME

Die Anweisung NAME definiert den Namen des ausführbaren Moduls der Anwendung. Der Modulname identifiziert das Modul, wenn Funktionen daraus exportiert werden. Bei 32-Bit-Anwendungen muß NAME vor EXETYPE stehen. Falls die Anweisungen NAME und EXETYPE nicht denselben Typ für das zu erzeugende Programm angeben, hat die Angabe bei NAME Vorrang. Die Syntax der Anweisung NAME lautet:

```
NAME ModulName [WINDOWSAPI] | [WINDOWCOMPAT]
```

ModulName (optional) besteht aus einem oder mehreren ASCII-Zeichen (Großbuchstaben), die den Namen des ausführbaren Moduls bilden. Der Modulname muß dem Namen der ausführbaren Datei entsprechen. So hat z.B. eine Anwendung mit der ausführbaren Datei SAMPLE.EXE den Modulnamen SAMPLE.

Wenn der Parameter *ModulName* nicht angegeben ist, nimmt TLINK an, daß der Modulname dem Namen der ausführbaren Datei entspricht. Wenn Sie keinen Modulnamen angeben und die ausführbare Datei zum Beispiel MYAPP.EXE heißt, setzt TLINK voraus, daß der Modulname MYAPP ist.

Enthält die Moduldefinitionsdatei weder eine NAME- noch eine LIBRARY-Anweisung, so verhält sich TLINK wie bei einer NAME-Anweisung ohne den Parameter *ModulName*.

WINDOWAPI bezeichnet ein ausführbares Windows-Programm, die Angabe ist gleichwertig mit der Option **/aa** von TLINK32.

WINDOWCOMPAT bezeichnet ein Windows-kompatibles Textbildschirm-Programm, die Angabe ist gleichwertig mit der Option **/ap** von TLINK32.

Anweisung SEGMENTS

Die Anweisung SEGMENTS definiert die Segmentattribute von zusätzlichen Code- und Datensegmenten. Die Syntax lautet:

```
SEGMENTS
SegmentName [CLASS 'KlassenName'] [MinAlloc]
     [SHARED | NONSHARED]
     [PRELOAD | LOADONCALL]
```

- *SegmentName* ist ein String, der dem neuen Segment einen Namen gibt. Dabei kann es sich um einen beliebigen Namen handeln, einschließlich

der Namen _TEXT und _DATA, die die Standard-Code- und Datensegmente bezeichnen.

- *KlassenName* (optional) ist der Klassenname des angegebenen Segments. Ist kein Klassenname angegeben, verwendet TLINK den Klassennamen CODE.
- *MinAlloc* (optional) ist ein Integerwert, der die minimale Größe des zu reservierenden Speicherplatzes für das Segment angibt. TLINK und TLINK32 ignorieren diese Angabe.
- SHARED (Voreinstellung für 16-Bit-DLLs) bedeutet, daß alle Prozesse auf dieselbe Kopie des Datensegments zugreifen. NONSHARED (Voreinstellung für Programme und 32-Bit-DLLs) bedeutet, daß für jeden Prozeß, der das Datensegment benutzen muß, eine eigene Kopie des Datensegments geladen wird.
- PRELOAD bedeutet, daß das Segment sofort geladen wird. LOADONCALL (wird von TLINK32 ignoriert) bedeutet, daß es erst dann geladen wird, wenn auf Daten darin zugegriffen oder eine Funktion darin aufgerufen wird. Der Ressourcen-Compiler beachtet die Option LOADONCALL nicht unbedingt und lädt dann stattdessen Segmente im voraus (wie PRELOAD).

Anweisung STACKSIZE

Die Anweisung STACKSIZE definiert die Anzahl Bytes, die die Anwendung für ihren lokalen Stack benötigt. Eine Anwendung benutzt den lokalen Stack immer dann, wenn sie Funktionen aufruft. Für DLLs dürfen Sie die Anweisung STACKSIZE nicht verwenden.

Die Syntax lautet:

```
STACKSIZE Bytes
```

Bytes ist ein Integerwert, der die Stack-Größe in Bytes angibt.

Anweisung STUB

Die Anweisung STUB fügt eine durch *Dateiname* definierte ausführbare DOS-Datei am Anfang des Moduls ein. Der ausführbare Stub zeigt eine Warnung an und bricht dann die Ausführung ab, falls der Benutzer versucht, das Programm mit dem Stub in der falschen Umgebung auszuführen (zum Beispiel beim Starten eines Windows-Programms unter DOS).

Borland C++ fügt am Beginn einer Windows-Anwendung einen vordefinierten Stub ein, sofern kein anderer Stub mit der Anweisung STUB definiert wurde. Verwenden Sie deshalb nicht STUB, um lediglich WINSTUB.EXE einzubinden, weil der Linker dies automatisch tut. Die Syntax lautet:

```
STUB "Dateiname"
```

Dateiname gibt den Namen der ausführbaren DOS-Datei an, die am Beginn des Moduls eingefügt werden soll. Der Name muß den DOS-Konventionen für Dateinamen entsprechen.

Befindet sich die mit *Dateiname* spezifizierte Datei nicht im aktuellen Verzeichnis, sucht TLINK in den Verzeichnissen, die durch die Umgebungsvariable PATH festgelegt sind.

Voreinstellungen für Moduldefinitionsdateien

Eine Moduldefinitionsdatei ist bei Borland C++ für die Erzeugung einer ausführbaren Windows-Datei nicht zwingend erforderlich.

Wenn Sie keine Moduldefinitionsdatei angeben, gelten folgende Voreinstellungen:

```
CODE          PRELOAD  MOVEABLE  DISCARDABLE
DATA          PRELOAD  MOVEABLE  MULTIPLE (für Anwendungen)
              PRELOAD  MOVEABLE  SINGLE (für DLLs)
HEAPSIZE      4096
STACKSIZE     5120 (1048576 bei TLINK32)
```

Um die Anweisung EXETYPE zu ersetzen, kann der Linker von Borland C++ anhand der Einstellungen in der IDE oder der Optionen in der Kommandozeile die Art der zu erzeugenden ausführbaren Datei erkennen.

Als Ersatz für den IMPORTS-Abschnitt der Moduldefinitionsdatei können Sie eine Importbibliothek aufnehmen.

Wenn Sie in Ihrem C- oder C++ Quelltext das Schlüsselwort **_export** in den Definitionen von Export-Funktionen einsetzen, benötigen Sie den Abschnitt EXPORTS nicht. Beachten Sie jedoch, daß, wenn Sie _export zum Exportieren einer Funktion verwenden, diese Funktion über ihren Namen und nicht über ihren Ordinalwert exportiert wird (das Exportieren über den Ordinalwert ist normalerweise effizienter).

Wenn Sie verschiedene Attribute im Vergleich zu den Voreinstellungen ändern möchten, müssen Sie eine Moduldefinitionsdatei verwenden.

Anweisung HEAPSIZE

Die Anweisung HEAPSIZE definiert die Anzahl der Bytes, die von der Anwendung für dessen lokalen Heap benötigt wird. Eine Anwendung benutzt den lokalen Heap, wenn lokaler Speicher alloziert wird. Die Syntax für HEAPSIZE lautet:

HEAPSIZE *Bytes*

Bytes ist ein ganzzahliger Wert, der die Heap-Größe in Bytes festlegt. Für 16-Bit-Anwendungen darf diese Zahl 65.535 (physikalische Segmentgröße) nicht überschreiten.

TLINK32 ignoriert jeden STACKSIZE- oder HEAPSIZE-Wert, der kleiner als 64K ist. Dennoch kann dies auf der Befehlszeile außer Kraft gesetzt werden.

TLINK32 unterstützt die zusätzliche Syntax:

HEAPSIZE Reserve,Festlegen

STACKSIZE Reserve,Festlegen

Mit dieser Syntax wird durch das Feld *Reserve* die maximale Speichergröße sowohl des Stacks als auch des Heaps festgelegt. Das Feld *Festlegen* bestimmt, wieviel Speicher das Betriebssystem während der Lade- und Initialisierungszeit einer Anwendung zur Verfügung stellen muß.

K A P I T E L 10

Arbeiten mit den Ressourcen-Dienstprogrammen

Dieses Kapitel beschreibt die Ressourcen-Dienstprogramme von Borland.

- BRCC.EXE und BRCC32.EXE sind die Ressourcencompiler von Borland. Sie compilieren Ressourcen-Scriptdateien (.RC-Dateien) und erzeugen daraus eine binäre .RES-Datei.
- RLINK.EXE und RLINK32.DLL (über TLINK32.EXE) sind die Ressourcen-Linker von Borland. Sie linken Ressourcen in Form von .RES-Dateien zu einer .EXE-Datei hinzu und kennzeichnen die resultierende .EXE-Datei als ausführbar unter Windows.
- BRC.EXE und BRC32.EXE sind Bedienungsoberflächen, von denen aus Sie BRCC oder BRCC32 und RLINK oder RLINK32 (über TLINK32) in einem einzigen Schritt starten können.

Alle 32-Bit-Ressourcen-Dienstprogramme arbeiten exakt so wie die 16-Bit-Versionen, soweit in diesem Kapitel nichts anderes angegeben ist.

Windows-Anwendungsprogramme haben alle eine bedienungsfreundliche und standardisierte Benutzerschnittstelle. Deren Teile werden als Ressourcen bezeichnet. Dazu gehören:

- Menüs
- Dialogfenster
- Zeiger
- Symbole
- Bitmaps
- Strings
- Tastenkürzel
- Schriften

Ressourcen werden außerhalb Ihres Quellprogramms definiert und dann beim Linkvorgang der ausführbaren Datei hinzugefügt. Das Anwendungsprogramm lädt die Ressourcen nur dann in den Speicher, wenn sie benötigt werden; dadurch wird der Speicherbedarf gering gehalten.

Ressourcen-Scriptdateien (.RC-Dateien) sind Textdateien, die die Ressourcen beschreiben, die ein bestimmtes Anwendungsprogramm benutzen wird. BRCC und RC lesen die .RC-Datei, um die Ressourcen in eine binäre Ressourcendatei (.RES) zu compilieren. RLINK fügt dann die .RES-Datei, die Ihre Ressourcen enthält, Ihrer ausführbaren Datei hinzu (dieser Vorgang wird Ressourcelinken genannt).

BRCC.EXE: Der Ressourcencompiler

BRCC ist die Kommandozeilenversion des Ressourcencompiler von *Resource Workshop*. Er liest eine Ressourcen-Scriptdatei (.RC) und erzeugt eine Ressourcen-Objektdatei (.RES) als Ausgabe. BRCC arbeitet mit der folgenden Kommandosyntax:

```
BRCC [Optionen] <Dateiname>
```

Tabelle 10.1 zeigt alle Optionen für BRCC. Beachten Sie, daß die Groß- und Kleinschreibung bei den Optionen bedeutungslos ist (-r ist gleichwertig mit -R).

Tabelle 10.1 BRCC (Ressourcen-compiler von Borland

Option	Beschreibung
Befehle werden aus der angegebenen Datei entnommen.	
-d<Name><Strg>	Definiert ein Symbol für den Präprozessor.
-fo<Dateiname>	Benennt die .RES-Datei in *Dateiname* um. (Standardmäßig erzeugt BRCC eine .RES-Datei mit demselben Namen wie die RC-Eingabedatei).
-i<Pfad>	Fügt dem Include-Suchpfad ein oder mehrere Verzeichnisnamen (getrennt durch Semikolons) hinzu.
-r	Diese Option wird ignoriert, sie ist nur zur Kompatibilität mit anderen Ressourcencompilern vorhanden.
-v	Zeigt alle Meldungen des Compiler während des Compilierens im Klartext an.
-x	Verwirft den aktuellen Include-Pfad.
-? oder -h	Zeigt Hilfe-Informationen zu den Optionen an.
-30	Erzeugt Windows-3.0-kompatible .RES-Dateien.
-31	Erzeugt Windows-3.1-kompatible .RES-Dateien.
-w32	Erzeugt Win32-kompatible RES-Dateien.

Wie der Ressourcencompiler des *Resource Workshop* sind auch bei BRCC allgemeine ressourcenbezogene Windows-Konstanten wie zum Beispiel WS_VISIBLE und BS_PUSHBUTTON vordefiniert. Außerdem sind auch zwei speziell auf den Compiler bezogene Symbole definiert: RC_INVOKED und WORKSHOP_INVOKED. Verwenden Sie diese Symbole im Quelltext in Verbindung mit Präprozessor-#if-Direktiven, um die Compilierung zu steuern. Mit den folgenden Anweisungen läßt sich zum Beispiel die Compilierungsgeschwindigkeit wesentlich erhöhen:

```
#ifndef WORKSHOP_INVOKED
#include <windows.h>
#endif
```

Im folgenden Beispiel werden dem Include-Pfad zwei Verzeichnisse hinzugefügt, und es wird eine .RES-Datei mit demselben Namen wie die .RC-Eingabedatei erzeugt:

```
brcc -i<dir1>;<dir2> <Dateiname>.RC
```

Das nächste Beispiel ergibt eine .RES-Ausgabedatei mit einem von der .RC-Eingabedatei verschiedenen Namen:

```
brcc -fo<Dateiname>.RES <Dateiname>.RC
```

RLINK: Der Ressourcen-Linker

RLINK verbindet eine .RES-Datei mit einer .EXE-Datei zu einer neuen, unter Windows ablauffähigen Datei. RLINK benutzt als Eingabe eine oder mehrere Objektdateien (.RES) und eine einzige Datei mit einem ausführbaren Windows-Programm. RLINK linkt die Ressourcen, indem er Stringtabellen und Botschaftstabellen vervollständigt und dann die so bearbeiteten Ressourcen mit dem ausführbaren Programm verbindet.

Die Befehlssyntax von RLINK lautet wie folgt:

```
rlink [Optionen] <Dateiname>.RES <Dateiname>.EXE
```

RLINK kennt folgende Optionen:

Tabelle 10.2: Optionen von RLINK

Option	Beschreibung
@<Dateiname>	Befehle werden aus der angegebenen Datei entnommen.
-d	Entfernt Ressourcen aus der .EXE-Datei (keine .RES-Datei angeben)
-fe<Dateiname>	Benennt die Ausgabedatei (.EXE) um.
-fi<Dateiname>	Benennt die .EXE-Eingabedatei um.
-r	Segmente nicht für schnelleres Laden umordnen.
-v	Zeigt alle Meldungen des Linkers während des Linkens im Klartext an.
-vx	Erzeugt eine Liste der Ressourcen, verbindet sie aber nicht mit der .EXE-Datei.
-? oder -h	Zeigt Hilfeinformationen zu den Optionen an.
-30	Erzeugt Windows-3.0-kompatible .RES-Dateien.
-31	Erzeugt Windows-3.1-kompatible .RES-Dateien.

Im folgenden Beispiel werden die Ressourcen der .RES-Datei gelinkt und zu der .EXE-Datei hinzugefügt:

```
rlink <Dateiname>.RES <Dateiname>.EXE
```

In diesem Beispiel werden die Ressourcen aus zwei .RES-Dateien gelinkt und zu der .EXE-Datei hinzugefügt:

```
rlink -fi<Dateiname>.RES <Dateiname>.RES <Dateiname>.EXE
```

Im nächsten Beispiel wird der Programmcode in der .EXE-Eingangsdatei mit den Ressourcen in der .RES-Eingangsdatei verbunden und eine .EXE-Ausgangsdatei mit einem neuen Namen erzeugt.

```
rlink -fe<Dateiname>.EXE <Dateiname>.RES <Dateiname>.EXE
```

Im letzten Beispiel entnimmt RLINK seine Befehle einer Befehlsdatei (.RLK) entnommen. Dann werden die Ressourcen aus den drei .RES-Dateien gelinkt und mit der EXE-Datei verbunden:

```
rlink @<Dateiname>.RLK
```

Die Kommandodatei (<Dateiname>.RLK) hat dabei folgenden Inhalt:

```
-fi<Dateiname>.RES
-fi<Dateiname>.RES
<Dateiname>.RES
<Dateiname>.EXE
```

BRC.EXE: Die Bedienungsoberfläche für die Ressourcen-Dienstprogramme

Der Ressourcencompiler (BRC) von Borland ist praktisch eine Bedienungsoberfläche für Compiler und Linker. Er ruft je nach der angegebenen Kommandozeile entweder BRCC oder RLINK (oder beide) auf. Die Kommandozeilen-Syntax von BRC lautet wie folgt:

```
brc [Optionen] <Dateiname>.RC [<Dateiname>.EXE]
```

BRC kennt folgende Optionen:

Tabelle 10.3
Optionen von BRC

Option	Beschreibung
<Name>=String	Definiert ein Symbol, das mit der Präprozessor-Direktive #ifdef abgefragt werden kann.
-fo<Dateiname>	Benennt die .RES-Datei um.
-fe<Dateiname>	Benennt die .EXE-Datei um.
-fi<Dateiname>	Angabe weiterer .RES-Dateien.
-i<Pfad>	Fügt dem Include-Suchpfad ein oder mehrere Verzeichnisnamen (getrennt durch Semikolons) hinzu.
-k	Keinen Bereich für schnelleres Laden erzeugen.
-r	Erzeugt lediglich eine .RES-Datei; die compilierte .RES-Datei wird der .EXE-Datei nicht hinzugefügt.
-v	Zeigt alle Meldungen während des Compilierens und Linkens im Klartext an.
-x	Veranlaßt den Compiler, die Umgebungsvariable INCLUDE zu ignorieren, während er nach Include- oder Ressourcendateien sucht.
-31	Erzeugt Windows-3.1-kompatible .RES-Dateien.
-w32	Erzeugt Win32-kompatible .RES-Dateien.

Je nach Art Ihrer Aufgabe können Sie mehrere Variationen der einfachen Kommandozeilen-Syntax verwenden:

- Der folgende Befehl bewirkt, daß die .RC-Datei compiliert, eine .RES-Datei erzeugt und dann die .RES-Datei der ausführbaren Datei hinzugefügt wird:

```
brc <Dateiname>.RC [<Dateiname>.EXE]
```

BRC sucht automatisch nach einer .EXE-Datei mit demselben Namen wie die .RC-Datei. Sie müssen deshalb den Namen der .EXE-Datei nur dann angeben, wenn er vom Namen der .RC-Datei abweicht.

- Der folgende Befehl bewirkt, daß eine .RES-Datei, aber keine .EXE-Datei erzeugt wird. Wenn Sie in der Kommandozeile den Namen einer .EXE-Datei angeben, ignoriert BRC diesen:

```
brc -r<Dateiname>.RC
```

- Der nächste Befehl bewirkt, daß eine vorhandene .RES-Datei einer ausführbaren Datei hinzugefügt wird. Den Name der .EXE-Datei müssen Sie nur dann angeben, wenn er sich von dem der .RES-Datei unterscheidet.

```
brc <Dateiname>.RES [<Dateiname>.EXE]
```

K A P I T E L 11

Bibliotheken

In diesem Kapitel werden mehrere Kommandozeilen-Tools und Bereiche der IDE vorgestellt, die für die Arbeit mit Bibliotheken erforderlich sind:

- IMPLIB erzeugt Importbibliotheken, und IMPDEF erzeugt Moduldefinitionsdateien (.DEF-Dateien). Importbibliotheken und Moduldefinitionsdateien informieren den Linker über Funktionen, die aus dynamischen Linkbibliotheken (DLLs) importiert werden.
- Mit TLIB werden Bibliotheken einzelner OBJ-Dateien (Objektmodule) verwaltet. In einer Bibliothek können mehrere Objektmodule bequem als eine Einheit verwaltet werden.

IMPLIB: Verwaltung von Importbibliotheken

IMPLIB erzeugt Importbibliotheken. IMPLIB übernimmt DLLs, Moduldefinitionsdateien oder beides als Eingabe und erzeugt daraus eine Importbibliothek. Die IDE verwendet IMPLIB als Übersetzer für eine DLL-Zieldatei. (In Kapitel 2 finden Sie Informationen über die Projektverwaltung und Zieldateien.) Wenn Sie eine DLL als Zieldatei hinzufügen, dann compiliert und linkt die Projektverwaltung alle abhängigen Dateien und erzeugt eine DLL-Datei. Danach wird IMPLIB ausgeführt und eine LIB-Datei erzeugt. IMPLIB kann auch in der IDE ausgeführt werden. (In Kapitel 1 finden Sie Informationen über den Einsatz von Tools innerhalb der IDE.)

Importbibliotheken bestehen aus Records. Jeder Record enthält den Namen einer DLL und die Position der importierbaren Funktionen innerhalb der DLL. Diese Records werden von TLINK oder vom IDE-Linker zur Anwendung hinzugebunden und stellen Windows die Informationen zur Verfügung, die es zur Auflösung von DLL-Funktionsaufrufen benötigt. Eine Importbibliothek kann zum Teil oder im ganzen durch den IMPORTS-Abschnitt einer Moduldefinitionsdatei ersetzt werden.

Wenn Sie bereits eine Windows-Anwendung erzeugt haben, so haben Sie auch schon mit mindestens einer Importbibliothek gearbeitet: IMPORT.LIB

ist die Importbibliothek für die Standard-DLLs von Windows. (IMPORT.LIB wird beim Erzeugen von Windows-Anwendungen in der IDE oder mit BCC automatisch hinzugelinkt. Wenn Sie Ihre Anwendungen separat mit TLINK linken, so müssen Sie IMPORT.LIB explizit angeben.)

Eine Importbibliothek listet einige oder alle exportierten Funktionen einer oder mehrerer DLLs auf. IMPLIB erzeugt eine Importbibliothek direkt aus DLLs oder aus Moduldefinitionsdateien für DLLs (oder aus beidem).

Sie können eine Importbibliothek folgendermaßen erzeugen:

```
IMPLIB Optionen BibName [DefDateien... | DLLs...]
```

Eine DLL kann auch die Erweiterung .EXE oder .DRV haben.

Optionen ist eine optionale Liste einer oder mehrerer IMPLIB-Optionen (siehe Tabelle 11.1), *BibName* ist der Name der neuen Importbibliothek, *DefDateien* ist eine Liste einer oder mehrerer existierender Moduldefinitionsdateien für eine oder mehrere DLLs, und *DLLs* ist eine Liste einer oder mehrerer existierender DLLs. Mindestens eine DLL oder Moduldefinitionsdatei muß angegeben werden.

Tabelle 11.1: IMPLIB-Optionen

Option	Beschreibung
-c	Unterscheidung von Groß-/Kleinschreibung bei Symbolen. Wenn Sie zwei Symbole verwenden, die sich lediglich in der Groß-/Kleinschreibung unterscheiden (wie zum Beispiel *MYSYM* und *mysym*) und die Option **-c** nicht verwenden, dann unterscheidet IMPLIB die beiden Symbole nicht.
-i	IMPLIB wird angewiesen, WEP zu ignorieren. WEP ist die Windows-Exit-Prozedur, die zum Beenden einer DLL benötigt wird. Verwenden Sie diese Option, wenn Sie beim Aufruf von IMPLIB mehrere Linkbibliotheken angegeben haben.
-w	Warnungen werden unterdrückt.

Optionen müssen in Kleinbuchstaben angegeben und entweder mit einem Schräg- oder einem Bindestrich begonnen werden.

IMPDEF: Verwaltung von Moduldefinitionsdateien

Importbibliotheken ermöglichen den Zugriff auf die Funktionen in Windows-DLLs. Im vorherigen Absatz finden Sie hierzu weitere Details.

IMPDEF übernimmt einen DLL-Namen als Eingabe und erzeugt daraus eine Moduldefinitionsdatei mit einem EXPORTS-Abschnitt, der die Namen der von der DLL exportierten Funktionen enthält. Die Aufrufsyntax lautet:

```
IMPDEF Zielname.DEF Quellname.DLL
```

Aus der Datei *Quellname*.DLL wird durch den obigen Aufruf eine Moduldefinitionsdatei mit dem Namen *Zielname*.DEF erzeugt. Die Moduldefinitionsdatei sieht folgendermaßen aus:

```
LIBRARY      Dateiname
DESCRIPTION  'Beschreibung'
EXPORTS
             ExportFunkName        @Ordinal
                 :
             ExportFunkName        @Ordinal
```

Dateiname ist dabei der Pfadname der DLL, *Beschreibung* ist der Wert der DESCRIPTION-Anweisung, sofern die Linkbibliothek zuvor mit einer Moduldefinitionsdatei gelinkt wurde, *ExportFunkName* bezeichnet eine exportierte Funktion, und *Ordinal* ist der Ordinalwert (ein Integer) dieser Funktion.

Klassen in einer DLL

Der Einsatz von IMPDEF empfiehlt sich für eine DLL, die C++ Klassen beinhaltet. Wenn Sie das Schlüsselwort **_export** bei der Klassendefinition verwenden, werden alle Nicht-Inline-Elementfunktionen und alle statischen Datenelemente der Klasse exportiert. In diesem Fall ist es einfacher, eine Moduldefinitionsdatei von IMPDEF anlegen zu lassen, da diese alle exportierten Funktionen auflistet und die Elementfunktionen wie auch die statischen Datenelemente automatisch aufgenommen werden.

Da die Namen dieser Funktionen abgekürzt werden, wäre es sehr mühsam, sie alle im EXPORTS-Abschnitt einer Moduldefinitionsdatei aufzulisten, um daraus dann eine Importbibliothek zu erzeugen. Wenn Sie eine Moduldefinitionsdatei von IMPDEF erzeugen lassen, enthält diese den Ordinalwert jeder exportierten Funktion. Wenn der exportierte Name abgekürzt ist, setzt IMPDEF auch den ursprünglichen Namen der Funktion als Kommentar hinter den Funktionseintrag. Eine von IMPDEF angelegte Moduldefinitionsdatei für eine DLL mit C++ Klassen würde zum Beispiel wie folgt aussehen:

```
LIBRARY      Dateiname

DESCRIPTION  'Beschreibung'

EXPORTS      AbgekürzterExportFunkName   @Ordinal   ; ExportFunkName
                 :
EXPORTS      AbgekürzterExportFunkName   @Ordinal   ; ExportFunkName
```

Dateiname ist dabei der Pfadname der DLL, *Beschreibung* ist der Wert der DESCRIPTION-Anweisung, sofern die Linkbibliothek zuvor mit einer Moduldefinitionsdatei gelinkt wurde, *AbgekürzterExportFunkName* bezeichnet den abgekürzten Funktionsnamen, *Ordinal* ist der Ordinalwert (ein Integer) der Funktion, und *ExportFunkName* ist der ursprüngliche Funktionsname.

Funktionen in einer DLL

IMPDEF legt eine editierbare Quelltextdatei an, die alle exportierten Funktionen der DLL enthält. Sie können diese DEF-Datei so editieren, daß sie nur diejenigen Funktionen enthält, die Sie einer bestimmten Anwendung zur Verfügung stellen wollen. Danach rufen Sie IMPLIB für die editierte DEF-Datei auf und erhalten als Ergebnis eine Importbibliothek, die die Importinformationen für eine bestimmte Teilmenge der exportierten Funktionen einer DLL enthält.

Nehmen wir beispielsweise an, Sie wollen eine DLL erstellen, die von verschiedenen Anwendungen benutzt werden kann. Jede exportierte Funktion in den DLLs wird mit dem Schlüsselwort **_export** definiert. Wenn nun alle Anwendungen auch alle exportierten Funktionen der Linkbibliothek verwenden würden, dann bräuchten Sie lediglich IMPLIB einzusetzen, um für die Linkbibliothek eine Importbibliothek anzulegen, die Sie dann zusammen mit der Linkbibliothek verwenden würden. Diese Importbibliothek würde die Importinformationen für alle exportierten Funktionen der DLL enthalten und könnte zu jeder Anwendung gelinkt werden. Dadurch entfällt die Notwendigkeit, für jede Anwendung die von ihr benutzten DLL-Funktionen im IMPORTS-Abschnitt ihrer Moduldefinitionsdatei aufzulisten.

Wenn Sie nun aber einer bestimmten Anwendung nur einige der exportierten Funktionen der DLL zur Verfügung stellen wollen, dann sollten Sie im Idealfall eine angepaßte Importbibliothek zur Anwendung hinzulinken, die nur die Importinformationen für die Teilmenge der DLL-Funktionen enthält, die von der Anwendung auch tatsächlich benutzt werden. Alle übrigen exportierten DLL-Funktionen werden dadurch vor dieser Anwendung verborgen.

Zur Erzeugung einer solchen Importbibliothek wenden Sie IMPDEF auf die compilierte und gelinkte DLL an. IMPDEF generiert eine Moduldefinitionsdatei, die im EXPORTS-Abschnitt alle exportierten DLL-Funktionen auflistet. Sie können diese Moduldefinitionsdatei editieren und dabei alle Einträge im EXPORTS-Abschnitt entfernen, die in der angepaßten Importbibliothek nicht erscheinen sollen. Danach rufen Sie IMPLIB für diese Importbibliothek auf. Als Ergebnis erhalten Sie eine Importbibliothek, die Importinformationen nur für diejenigen exportierten Funktionen enthält, die im EXPORTS-Abschnitt der Moduldefinitionsdatei aufgelistet wurden.

TLIB: Bibliotheksverwaltung

Bei der Modifikation einer vorhandenen Bibliothek legt TLIB stets eine Sicherungskopie der ursprünglichen Bibliotheksdatei mit der Erweiterung .BAK an.

Die zu Borland C++ gehörenden Bibliotheken wurden mit TLIB erzeugt. Mit TLIB können Sie Ihre eigenen Bibliotheken erzeugen und verändern, Sie können aber auch die Bibliotheken von Borland C++ oder solche, die Sie von anderen Programmierern erhalten oder die Sie gekauft haben, verändern. Mit TLIB können Sie

- neue Bibliotheken aus mehreren Objektmodulen erstellen;
- Objektmodule oder andere Bibliotheken zu einer vorhandenen Bibliothek hinzufügen;
- Objektmodule aus einer vorhandenen Bibliothek löschen;
- Objektmodule in einer vorhandenen Bibliothek ersetzen;
- Objektmodule aus einer vorhandenen Bibliothek extrahieren;
- den Inhalt einer neuen oder vorhandenen Bibliothek auflisten.

Siehe den nachfolgenden Abschnitt über die Option **/E**.

TLIB kann auch ein erweitertes Verzeichnis erzeugen (und in die Bibliotheksdatei einbinden), das die Linkgeschwindigkeit erhöht.

TLIB ist für die Erstellung von ausführbaren Programmen mit Borland C++ zwar nicht zwingend erforderlich, beschleunigt jedoch die Entwicklung größerer Projekte.

Gründe für den Einsatz von Objektmodul-Bibliotheken

Bei der Programmierung in C und C++ werden Sie eine Vielzahl nützlicher Funktionen und Klassen erstellen. Wegen der Modularität von C und C++ werden Sie diese Funktionen wahrscheinlich auf mehrere separat compilierte Quelltextdateien verteilen. Ein bestimmtes Programm benutzt eventuell nur eine Teilmenge der insgesamt vorhandenen Funktionen.

Eine Objektmodul-Bibliothek verwaltet eine Sammlung von Funktionen und Klassen. Wenn Sie eine Bibliothek zu Ihrem Programm hinzulinken, durchsucht der Linker die Bibliothek und wählt automatisch nur diejenigen Module aus, die für das aktuelle Programm benötigt werden.

Die TLIB-Kommandozeile

Die TLIB-Kommandozeile hat folgende allgemeine Form (optionale Elemente sind in eckigen Klammern aufgeführt):

```
tlib [@Response-Datei] [Option] BibName [Operationen] [, Listendatei]
```

Sie können eine Bibliothek erzeugen, indem Sie sie in der IDE als Ziel in einer Projektdatei angeben. Wählen Sie bei Verwendung von TargetExpert als Zieltyp *Statische Bibliothek* aus. (In Kapitel 2 finden Sie eine Anleitung zur Handhabung des Projekt-Managers.) TLIB ist der Standard-Übersetzer für eine Bibliotheksdatei und verwendet Optionen, die Sie im

Dialogfenster *Projekt Optionen* im Thema *Bibliotheksverwaltung* einstellen. (Wählen Sie dazu *Optionen/Projekt* aus dem Hauptmenü.)

Tabelle 11.2: TLIB-Optionen

TLIB-Option	IDE-Option	Beschreibung
@Response-Datei		Pfad und Name der Response-Datei, die Sie verwenden möchten. Sie können mehrere Response-Dateien angeben.
BibName		Pfad und Name der Bibliothek, die Sie erzeugen oder modifizieren wollen. Bei jedem TLIB-Aufruf muß *BibName* angegeben werden. Jokerzeichen sind nicht zugelassen. Wird keine Dateinamenserweiterung angegeben, verwendet TLIB die Erweiterung .LIB. Verwenden Sie nur die Namenserweiterung .LIB, da sowohl die Kommandozeilen-Compiler (BCC und BCC32) als auch die IDE Bibliotheksdateien an dieser Erweiterung erkennen. **Hinweis:** Existiert die Bibliotheksdatei nicht, so legt TLIB eine Bibliothek des angegebenen Namens an, sofern *Hinzufügen*-Operationen im TLIB-Aufruf enthalten sind.
/C	Bibliothek unterscheidet Groß-/Kleinschreibung	Flag zur Berücksichtigung von Groß-/Kleinschreibung. Normalerweise wird diese Option nicht verwendet. Nachfolgend finden Sie hierzu Detailinformationen.
/E	erweitertes Verzeichnis	Erzeugt ein erweitertes Verzeichnis. Nachfolgend finden Sie hierzu Detailinformationen.
/PGröße	Bibliothek-Seitengröße	Setzt die Seitengröße der Bibliothek auf den Wert *Größe*. Nachfolgend finden Sie hierzu Detailinformationen.
/0	Kommentare entfernen	Kommentar-Records werden aus der Bibliothek entfernt.
Operationen		Die Liste der von TLIB auszuführenden Operationen. Die Operationen können in beliebiger Reihenfolge erscheinen. Wenn Sie nur den Inhalt der Bibliothek auflisten möchten, dann geben Sie keine Operation an.
Listendatei	Listendatei erzeugen	Benennt die Datei, in der der Inhalt der Bibliothek aufgelistet wird. Vor *Listendatei* muß ein Komma angegeben werden. Wenn Sie keinen Dateinamen angeben, so wird auch keine Listendatei erzeugt. In der Liste werden alle Module in alphabetischer Reihenfolge aufgeführt. Die Standarderweiterung für die *Listendatei* lautet .LST. Durch Angabe von CON bzw. PRN können Sie die Liste auf dem Bildschirm bzw. Drucker ausgeben.

Response-Dateien

Wenn Sie eine Vielzahl von Operationen verwenden oder bestimmte Operationen immer wieder wiederholen müssen, dann sollten Sie möglicherweise Response-Dateien benutzen. Eine Response-Datei ist eine ASCII-Textdatei (die beispielsweise mit dem Borland-Editor erzeugt werden kann), die alle (oder einen Teil der) Befehle für einen TLIB-Aufruf enthält. Mit Hilfe von Response-Dateien können Sie TLIB-Befehle zusammenstellen, die aufgrund ihrer Größe nicht in eine Kommandozeile passen würden. Response-Dateien können:

- mehr als eine Textzeile enthalten. Ein **&**-Zeichen am Ende einer Zeile bedeutet, daß noch eine weitere Zeile folgt.

- wahlweise auch nur einen Teil der verwendeten TLIB-Befehle enthalten. Sie können einen Teil der Optionen direkt in der Kommandozeile, einen anderen Teil in der Response-Datei angeben.
- zusammen mit weiteren Response-Dateien in einer einzigen TLIB-Kommandozeile benutzt werden.

Groß-/Kleinschreibung: Die Option /CTLIB;Groß-/Kleinschreibung

TLIB verwaltet ein Verzeichnis aller öffentlichen Symbole der Bibliotheksmodule. Wenn Sie der Bibliothek ein Symbol hinzufügen, dann muß dieses Symbol eindeutig sein. Der Versuch, einer Bibliothek ein bereits bekanntes Symbol hinzuzufügen, scheitert – TLIB zeigt eine Fehlermeldung an und fügt das Symbol nicht hinzu.

Verwenden Sie die Option **/C** keinesfalls, wenn Sie Ihre Bibliotheken mit anderen Linkern benutzen oder sie an dritte weitergeben wollen.

Da einige Linker nicht zwischen Groß- und Kleinschreibung unterscheiden, lehnt TLIB alle Symbole ab, die sich nur in der Groß-/Kleinschreibung voneinander unterscheiden (so werden beispielsweise die Symbole *LOOKUP* und *lookup* als gleich angesehen). TLINK *kann* jedoch zwischen Groß- und Kleinschreibung unterscheiden, so daß Sie – solange Sie ausschließlich mit TLINK arbeiten – die Option **/C** beim Hinzufügen von Modulen verwenden können, deren Symbole sich (zumindest teilweise) nur in der Groß-/Kleinschreibung unterscheiden.

Ein erweitertes Verzeichnis erstellen: Die Option /E

Zur Beschleunigung des Linkens großer Bibliotheksdateien können Sie TLIB anweisen, ein erweitertes Verzeichnis zu erzeugen und es der Bibliotheksdatei hinzuzufügen. Dieses Verzeichnis enthält in kompakter Form Informationen, die nicht im Standardverzeichnis enthalten sind und mit deren Hilfe TLIB nicht benötigte Module schneller von der Bearbeitung ausschließen kann.

Zur Erzeugung eines erweiterten Verzeichnisses für eine Bibliothek, die Sie ändern möchten, verwenden Sie die Option **/E**, wenn Sie TLIB zum Hinzufügen, Löschen oder Ersetzen von Modulen aufrufen. Möchten Sie ein erweitertes Verzeichnis für eine Bibliothek anlegen, die Sie ansonsten nicht ändern wollen, so benutzen Sie ebenfalls die Option **/E**. Durch den Aufruf `TLIB /E mylib` wird beispielsweise der Bibliothek *mylib* ein erweitertes Verzeichnis hinzugefügt.

Wenn Sie die Option **/E** beim Hinzufügen eines Moduls verwenden, das eine C++ Klasse mit einer virtuellen Funktion enthält, so erhalten Sie die Fehlermeldung: `Library contains COMDEF records -- extended dictionary not created (Bibliothek enthält COMDEF-Records -- erweitertes Verzeichnis wurde nicht erzeugt).`

Die Seitengröße definieren: Die Option /PTLIB; Seitengröße

Jede DOS-Bibliotheksdatei enthält an ihrem Ende – hinter allen Objektmodulen – ein Verzeichnis. Dieses Verzeichnis enthält für jedes Bibliotheksmodul die 16-Bit Adresse dieses Moduls innerhalb der LIB-Datei. Die Ein-

heit der Adreßangabe entspricht dabei der Seitengröße der Bibliothek (standardmäßig sind dies 16 Bytes).

Die Seitengröße einer Bibliothek legt die maximale Gesamtgröße aller Objektmodule in der Bibliothek fest, die 65.536 Seiten nicht überschreiten darf. Die voreingestellte (minimale) Seitengröße von 16 Bytes erlaubt eine maximale Bibliotheksgröße von einem Megabyte. Für die Erzeugung größerer Bibliotheken müssen Sie die Seitengröße mit Hilfe der Option **/P** erhöhen. Die Seitengröße muß eine Potenz von 2 sein und darf nicht kleiner als 16 und nicht größer als 32.768 sein.

Alle Module einer Bibliothek beginnen an einer Seitengrenze. Beispielsweise werden bei einer Seitengröße von 32 Bytes (die nach der Standardeinstellung nächstgrößere Einstellung) durchschnittlich 16 Bytes pro Modul durch Füll-Bytes verschwendet. Bei Überschreiten der maximalen Bibliotheksgröße gibt TLIB eine Fehlermeldung aus und schlägt die Verwendung der Option **/P** mit der nächstgrößeren Seitengröße vor.

Kommentare entfernen: Die Option /0TLIB;Kommentare entfernen

Durch die Option **/0** werden alle Kommentar-Records entfernt und somit die Bibliotheksgröße verringert. Es können beispielsweise Informationen zum Debuggen oder zur Symbolanzeige in Ihrer Bibliothek enthalten sein, die Sie nicht mehr länger benötigen.

Die Liste der Operationen

Die Liste der Operationen beschreibt die Aktionen, die TLIB ausführen soll. Sie besteht aus einer Folge von Operationen, die nacheinander ausgeführt werden. Jede Operation besteht aus einem ein oder zwei Zeichen langen Aktionssymbol, dem ein Datei- oder Modulname folgt.TLIB;Modulnamen Vor oder nach dem Aktionssymbol und dem Datei- bzw. Modulnamen sind Whitespace-Zeichen erlaubt – innerhalb des Aktionssymbols oder eines Namens sind sie nicht zulässig.

Sie können beliebig viele Aktionen in der Kommandozeile angeben, solange Sie die von DOS vorgegebene maximale Zeilenlänge von 127 Zeichen nicht überschreiten. Die Reihenfolge der Operationen spielt dabei keine Rolle, da TLIB die Operationen stets in der folgenden feststehenden Reihenfolge abarbeitet:TLIB;Reihenfolge der Operationen

Bei Ersetzen eines Moduls wird es zuerst entfernt und dann hinzugefügt.

1. Alle Extrahieren-Operationen werden ausgeführt.
2. Alle Löschen-Operationen werden ausgeführt.
3. Alle Hinzufügen-Operationen werden ausgeführt.

In Datei- oder Modulnamen sind keine Jokerzeichen zugelassen.

TLIB geht bei der Bestimmung des Modulnamens vom angegebenen Dateinamen aus und ignoriert dabei alle Angaben bezüglich Laufwerk, Pfad und Namenserweiterung. TLIB geht dabei immer von sinnvollen Standardeinstellungen aus. Um beispielsweise ein Modul mit der Erweiterung

.OBJ aus dem aktuellen Verzeichnis hinzuzufügen, müssen Sie lediglich den Modulnamen ohne Pfad und Erweiterung angeben.

TLIB kennt drei Aktionssymbole (–, **+**, *****), die Sie einzeln oder zu Paaren kombiniert für insgesamt fünf verschiedene Operationen verwenden können. Bei Operationen, die durch ein Zeichenpaar ausgelöst werden, spielt die Reihenfolge der Zeichen keine Rolle. In der folgenden Tabelle werden die Symbole und die von Ihnen ausgelösten Aktionen aufgelistet:

Tabelle 11.3: TLIB-Aktionssymbole

Um eine neue Bibliothek anzulegen, fügen Sie Module einer nicht existierenden Bibliotheksdatei hinzu.

Da Sie ein Modul nicht direkt umbenennen können, extrahieren und löschen Sie es zunächst. Dann benennen Sie die soeben erzeugte Datei um, und fügen Sie diese wieder in die Bibliothek ein.

Aktionssymbol	Name	Beschreibung
+	**Hinzufügen**	TLIB fügt die angegebene Datei der Bibliothek hinzu. Wenn keine Namenserweiterung angegeben wird, nimmt TLIB .OBJ an. Handelt es sich bei der Datei selbst um eine Bibliothek mit der Erweiterung .LIB, so fügt TLIB alle Bibliotheksmodule der Zielbibliothek hinzu. Ist ein Modul in der Bibliothek bereits vorhanden, so erhalten Sie eine Fehlermeldung darüber, daß das Modul nicht hinzugefügt werden konnte.
–	**Löschen**	TLIB löscht das angegebene Modul aus der Bibliothek. Wird das Modul in der Bibliothek nicht gefunden, so gibt TLIB eine Fehlermeldung aus. Sie können zwar einen vollständigen Pfadnamen mit Laufwerk und Namenserweiterung angeben – TLIB verwendet jedoch nur den Modulnamen.
*****	**Extrahieren**	TLIB erzeugt die angegebene Datei durch Kopieren des entsprechenden Moduls aus der Bibliothek. Wird das Modul in der Bibliothek nicht gefunden, so gibt TLIB eine Fehlermeldung aus. Existiert die angegebene Datei bereits, so wird sie überschrieben.
-* ***-**	**Extrahieren & Löschen**	TLIB kopiert das angegebene Modul in die entsprechende Datei und löscht es danach aus der Bibliothek.
–+, **+–**	**Ersetzen**	TLIB ersetzt das angegebene Modul durch die entsprechende Datei.

Beispiele

Die folgenden einfachen Beispiele demonstrieren die verschiedenen Aktionen, die mit TLIB durchgeführt werden können:

- Zur Erzeugung der Bibliothek MYLIB.LIB mit den Modulen X.OBJ, Y.OBJ und Z.OBJ geben Sie folgende Kommandozeile ein: `tlib mylib +x +y +z`
- Zur Erzeugung der Bibliothek MYLIB.LIB und der Listendatei MYLIB.LST geben Sie folgende Kommandozeile ein: `tlib mylib +x +y +z, mylib.lst`

Um die Listendatei CS.LST für die vorhandene Bibliothek CS.LIB zu erzeugen, geben Sie folgende Kommandozeile ein:

```
tlib cs, cs.lst
```

- Um in MYLIB.LIB das Modul X.OBJ durch eine neue Version zu ersetzen, A.OBJ hinzuzufügen und Z.OBJ zu löschen, geben Sie folgende Kommandozeile ein: `tlib mylib -+x +a -z`

Um das Modul Y.OBJ aus MYLIB.LIB zu extrahieren und eine Listendatei MYLIB.LST zu erzeugen, geben Sie folgende Kommandozeile ein:

```
tlib mylib *y, mylib.lst
```

- Um eine neue Bibliothek namens ALPHA mit den Modulen A.OBJ, B.OBJ ... G.OBJ zu erzeugen, benutzen Sie eine Response-Datei:
 - Erstellen Sie zunächst die Textdatei ALPHA.RSP:

    ```
    +a.obj +b.obj +c.obj &
           +d.obj +e.obj +f.obj &
           +g.obj
    ```

 - Danach rufen Sie TLIB mit der folgenden Kommandozeile auf, in der zusätzlich die Listendatei ALPHA.LST erzeugt wird:

    ```
    tlib alpha @alpha.rsp, alpha.lstw
    ```

K A P I T E L 12

Mit MAKE arbeiten

MAKE.EXE ist ein über die Kommandozeile gesteuertes Dienstprogramm für die Projektverwaltung; es hilft Ihnen, schneller zu compilieren, indem nur diejenigen Dateien des Projekts neu übersetzt werden, die sich seit der letzten Compilierung geändert haben. (MAKER ist eine im Real-Modus lauffähige Version von MAKE.) Wenn Sie in der IDE arbeiten, sollten Sie den IDE-Projektmanager (siehe Kapitel 2) verwenden.

Dieses Kapitel behandelt folgende Themen:

- Arbeitsweise von MAKE
- MAKE-Makros verwenden
- Inhalt einer Make-Datei
- MAKE-Direktiven benutzen
- Explizite und implizite Regeln anwenden

Arbeitsweise von MAKE

MAKE entnimmt die Regeln dafür, welche Dateien erzeugt werden sollen und wie dies zu geschehen hat, einer Textdatei (in der Voreinstellung MAKEFILE oder MAKEFILE.MAK). Sie können MAKE beispielsweise anweisen, per Compiler eine EXE-Datei dann neu erzeugen zu lassen, wenn die entsprechenden CPP-Quelltextdateien (die die Programmteile für die EXE-Datei enthalten) jünger als die vorhandene EXE-Datei selbst sind. MAKE ist besonders nützlich, wenn Sie ein Programm aus mehreren Dateien aufbauen, weil MAKE nur diejenigen Dateien neu compiliert, die Sie seit der letzten Compilierung geändert haben.

Es gibt zwei Arten von Regeln, explizite und implizite, die MAKE sagen, welche Datei von welcher abhängig ist. MAKE vergleicht dann die Zeitangaben der in einer Regel angegebenen Dateien und stellt so fest, ob ein Kommando ausgeführt werden soll. (Normalerweise weisen die Kommandos MAKE an, welche Dateien neu compiliert oder gelinkt werden

müssen, aber es kann sich auch um praktisch jedes andere Betriebssystem-Kommando handeln.)

MAKE akzeptiert auch die Jokerzeichen ***** und **?**.

Die allgemeine Syntax zum Aufruf von MAKE lautet:

```
MAKE [Optionen...] [Ziel(e)]
```

Um von der Kommandozeile aus Hilfe für MAKE zu erhalten, geben Sie `MAKE -?` oder `MAKE -h` ein.

Dabei steht `Optionen` für Optionen, die die Arbeitsweise von MAKE beeinflussen, und `Ziel(e)` bezeichnet die in einer Make-Datei angegebenen Namen der Zieldatei(en), die MAKE aufbauen soll. Zwischen den Optionen und `MAKE` muß ein einzelnes Leerzeichen stehen, Optionen und Ziele werden ebenfalls durch Leerzeichen getrennt.

Wenn Sie in der Kommandozeile `MAKE` eingeben, so führt MAKE laut Voreinstellung folgende Aktionen aus:

1. MAKE sucht im aktuellen Verzeichnis nach einer Datei namens BUILTINS.MAK. Diese Datei enthält Regeln, die MAKE immer beachtet, falls Sie nicht die Option **-r** angegeben haben. Wenn MAKE die Datei im aktuellen Verzeichnis nicht findet, so sucht es sie in dem Verzeichnis, in dem MAKE.EXE steht. Nachdem BUILTINS.MAK geladen ist, sucht MAKE nach einer Datei namens MAKEFILE oder MAKEFILE.MAK. Wenn MAKE keine dieser beiden Dateien findet, gibt es eine Fehlermeldung aus.
2. Wenn MAKE eine Make-Datei findet, versucht es, *ausschließlich* die erste Zieldatei der Make-Datei aufzubauen (allerdings kann die erste Zielangabe den Aufbau anderer Zieldateien fordern). MAKE prüft für das erste Ziel die Uhrzeit- und Datumsangaben der Dateien, von denen das Ziel abhängig ist. Wenn diese Ausgangsdateien jünger sind als die Zieldatei, so führt MAKE die zur Aktualisierung der Zieldatei spezifizierten Befehle aus. Siehe den Abschnitt *Verwenden der Make-Dateien* für weitere Informationen über die Befehle in Make-Dateien.
3. Wenn eine Datei, von der die erste Zieldatei abhängig ist, an anderer Stelle in der Make-Datei selbst wiederum als Zieldatei angegeben ist, so prüft MAKE die gegenseitigen Abhängigkeiten und baut erst diese Datei auf, bevor es die erste Zieldatei aufbaut. Diese Art der Kettenbildung wird verkettete Abhängigkeitsprüfung genannt.
4. Wenn der Aufbau einer Zieldatei fehlschlägt, so löscht MAKE die gerade im Aufbau befindliche Zieldatei wieder. Verwenden Sie die Direktive **.precious**, wenn Sie möchten, daß MAKE eine Zieldatei trotz Auftreten eines Fehlers bestehen läßt.

Um den Ablauf von MAKE zu unterbrechen, betätigen Sie die Tasten [*Strg*] [*Untbr*] oder [*Strg*] [*C*].

BUILTINS.MAK

Die Datei BUILTINS.MAK enthält vordefinierte Regeln und -makros, die MAKE auswertet, bevor es die Make-Datei liest. Wenn MAKE die Datei BUILTINS ignorieren soll, geben Sie die Option **-r** an. Verwenden Sie BUILTINS.MAK für Kommandos und Makros, die jedesmal ausgeführt werden sollen, wenn Sie MAKE benutzen. Standardmäßig enthält BUILTINS.MAK folgenden Text:

```
#
# Borland C++ - (C) Copyright 1992 by Borland International
#
CC = BCC
AS = TASM
RC = RC
.asm.obj:
  $(AS) $(AFLAGS) $&.asm
.c.exe:
  $(CC) $(CFLAGS) $&.c
.c.obj:
  $(CC) $(CFLAGS) /c $&.c
.cpp.obj:
  $(CC) $(CPPFLAGS) /c $&.cpp
.rc.res:
  $(RC) $(RFLAGS) /r $&
.SUFFIXES: .exe .obj .asm .c .res .rc
```

Arbeiten mit TOUCH.EXE

Gelegentlich möchten Sie vielleicht eine Zieldatei neu compiliert oder aufgebaut haben, obwohl sie nicht verändert ist. Eine Methode, dies zu erreichen, besteht in der Verwendung des Dienstprogramms TOUCH. Mit TOUCH können Sie die Zeit- und Datumseinträge einer oder mehrerer Dateien auf die momentanen Systemwerte ändern und so die Dateien »neuer« machen als die von ihnen abhängigen.

Sie können MAKE dazu zwingen, eine Zieldatei neu aufzubauen, indem Sie TOUCH auf eine der Dateien, von denen die Zieldatei abhängt, anwenden. Um dies bei einer oder mehreren Dateien zu tun, geben Sie in der Kommandozeile folgendes ein:

Bei TOUCH können Sie die Jokerzeichen ***** und **?** verwenden.

```
touch dateiname [dateiname...]
```

TOUCH aktualisiert dann Erstellungsdatum und -uhrzeit der Datei(en).

Wichtig!

Bevor Sie mit TOUCH arbeiten, vergewissern Sie sich, daß Ihre interne Systemuhr korrekt gestellt ist. Falls dies nicht der Fall ist, werden TOUCH und MAKE nicht korrekt arbeiten.

Optionen für MAKE

Die Kommandozeilenoptionen von MAKE bestimmen seine Arbeitsweise. Geben Sie die Optionen mit vorangestelltem Bindestrich (-) oder Schrägstrich (/) ein, und beachten Sie, daß Groß- und Kleinschreibung bei den Op-

tionen unterschieden wird. Um zum Beispiel die Datei PROJEKTA.MAK als Make-Datei zu benutzen, geben Sie `MAKE -fPROJEKTA.MAK` ein (ein Leerzeichen hinter **-f** ist optional). Für viele der Kommandozeilenoptionen gibt es gleichwertige Direktiven zur Benutzung in der Make-Datei (für weitere Informationen über Direktiven siehe den Abschnitt *Mit MAKE-Direktiven arbeiten*).

Tabelle 12.1: Optionen für MAKE

Option	Beschreibung
-h oder **-?**	Zeigt einen Hilfetext für MAKE an. Voreingestellte Optionen sind mit einem nachfolgenden Pluszeichen (**+**)gekennzeichnet.
-D*makro*	Definiert *makro* als ein einziges Zeichen, so daß der Ausdruck **!ifdef** *makro* in einer Make-Datei den Wert *Wahr* ergibt.
[**-D**]*makro*=[*string*]	Definiert *makro* als *string*. Wenn *string* Leer- oder Tabulatorzeichen enthält, muß *string* in Anführungszeichen eingeschlossen werden. Die Zeichen **-D** sind optional.
I*verzeichnis*	Sucht Include-Dateien erst im aktuellen Verzeichnis, danach in *verzeichnis*.
-K	Läßt von MAKE angelegte temporäre Dateien bestehen (normalerweise werden sie von MAKE gelöscht). Siehe auch unter KEEP.
-N	Läßt MAKE wie das Gegenstück NMAKE (von Microsoft) arbeiten (für weitere Informationen siehe den Abschnitt nach dieser Tabelle).
-U*makro*	Löscht vorangegangene Definitionen von *makro*.
-W	Schreibt die aktuell spezifizierten Nicht-String-Optionen in die Datei MAKE.EXE, so daß sie zu Standardvoreinstellungen werden.
-f*dateiname*	Verwendet *dateiname* oder *dateiname*.MAK statt MAKEFILE (ein Leerzeichen hinter **-f** ist optional).
-a	Bewirkt eine Abhängigkeitsprüfung von Include-Dateien (auch von verschachtelten), von denen eine OBJ-Datei abhängt, und aktualisiert diese OBJ-Datei, falls eine der Include-Dateien verändert ist. Siehe auch **-c**.
-c	Verbirgt die Informationen zur automatischen Abhängigkeitsprüfung, so daß MAKE möglicherweise schneller arbeitet. Verwenden Sie diese Option zusammen mit **-a**. Verwenden Sie diese Option nicht, wenn MAKE selbst Include-Dateien verändert (z.B. wenn TOUCH in einer Make-Datei aufgerufen wird, oder wenn im Rahmen der Ausführung von MAKE Header- oder Include-Dateien erzeugt werden).
-d*verzeichnis*	Wird zusammen mit **-S** verwendet, um das Laufwerk und Verzeichnis anzugeben, in dem MAKE seine Auslagerungsdateien anlegt. Diese Option ist bei MAKER wirkungslos.
-e	Ignoriert ein Makro, dessen Name dem einer Umgebungsvariablen gleicht (MAKE benutzt die Umgebungsvariable statt des Makros).
-i	Ignoriert den Exit-Code aller von MAKE aufgerufenen Programme und setzt den Programmaufbauvorgang stets fort.
-m	Zeigt Erstellungsdatum und -uhrzeit jeder Datei an, die MAKE gerade bearbeitet.
-n	Zeigt die Kommandos an, führt sie aber nicht wirklich aus (Hilfe zum Testen einer Make-Datei).
-p	Zeigt alle Makrodefinitionen und impliziten Regeln vor der Ausführung der Make-Datei an.
-q	Gibt den Exit-Code Null zurück, wenn das Ziel aktuell ist, andernfalls einen Code ungleich Null (zur Verwendung in Batch-Dateien).
-r	Ignoriert alle in BUILTINS.MAK etwa vorhandenen Regeln.
-s	Unterdrückt die laufende Bildschirmanzeige der Befehle.
-S	Lagert MAKE während der Ausführung von DOS-Kommandos aus dem Speicher aus, wodurch Speicherplatz gespart und die Compilierung auch großer Module ermöglicht wird. Diese Option ist bei MAKER wirkungslos.

Optionen zu Standardvoreinstellungen machen

Die Option **-W** erlaubt es Ihnen, einige MAKE-Optionen zu Standardvoreinstellungen zu machen, so daß sie jedesmal aktiviert werden, wenn Sie MAKE aufrufen. Um dies zu erreichen, geben Sie ein:

```
make  -option[-]  [-option][-]  . . . -W
```

Beispiel: Wenn Sie `MAKE -m -W` eingeben, bekommen Sie stets Erstellungsdatum und -uhrzeit der Dateien angezeigt. Um diese Voreinstellung wieder rückgängig zu machen, geben Sie `MAKE -m- -W` ein. Geben Sie `J` ein, wenn MAKE Sie fragt, ob die Änderungen in MAKE.EXE gespeichert werden sollen.

Warnung!

Die Option **-W** funktioniert nicht, wenn das DOS-Programm SHARE läuft. Sie erhalten dann die Anzeige `Fatal: unable to open file MAKE.EXE`. Die Option **-W** arbeitet auch nicht gleichzeitig mit folgenden MAKE-Optionen:

- **-D***makro*
- **-f***dateiname*
- **-D***makro=string*
- **-?** oder **-h**
- **-d***verzeichnis*
- **-I***verzeichnis*
- **-U***makro*

Kompatibilität zu NMAKE von Microsoft

Verwenden Sie die Option **-N**, wenn Sie Make-Dateien benutzen wollen, die ursprünglich für das Programm NMAKE (von Microsoft) geschrieben worden sind. Wenn Sie **-N** verwenden, ergeben sich folgende Änderungen:

- MAKE interpretiert den Operator **<** wie den Operator **&&**: Als Response-Dateien werden temporäre Dateien benutzt, die später gelöscht werden. Um eine Datei zu erhalten, benutzen Sie die Kommandozeilenoption **-K** oder die Direktive KEEP in der Make-Datei.

Im Normalfall löscht MAKE die von ihm erzeugten temporären Dateien.

```
<<TMPDATEI.TXT!
text
:
!KEEP
```

Wenn Sie nicht wollen, daß eine temporäre Datei erhalten wird, geben Sie NOKEEP oder einfach nur den Namen der temporären Datei ein. Wenn Sie NOKEEP mit Angabe einer temporären Datei benutzen und danach MAKE mit der Option **-K** aufrufen, so löscht MAKE die temporäre Datei.

- Das Makro **$d** hat eine andere Wirkung. Verwenden Sie statt dessen **!ifdef** oder **!ifndef**.
- Bei Pfadangaben, die von Makros zurückgegeben werden, fehlt der abschließende Schrägstrich (\). Beispiel: Wenn das Makro `$()` normalerweise `C:\CPP\` zurückgibt, so gibt es bei aktiver Option **-N** nur `C:\CPP` zurück.
- Wenn keine passende **.suffixes**-Direktive vorhanden ist, sucht NMAKE vom Ende der Make-Datei rückwärts nach entsprechenden Regeln.
- Das Makro **$*** wird in einer impliziten Regel immer zu dem Namen der Zieldatei erweitert anstatt zum Namen der Datei, von der die Zieldatei abhängt.

Arbeiten mit Make-Dateien

Eine Make-Datei ist eine ASCII-Datei mit Anweisungen für MAKE.EXE. Das Programm MAKE nimmt an, daß die Make-Datei den Namen MAKEFILE oder MAKEFILE.MAK hat, es sei denn, Sie benutzen die Option **-f** (siehe Tabelle 12.1).

MAKE baut entweder die Zieldateien, die Sie in der MAKE-Kommandozeile angeben, oder *ausschließlich* die erste Zieldatei, die es in der Make-Datei findet, auf (zum Aufbau mehrerer Zieldateien siehe den folgenden Abschnitt *Symbolische Zielangaben*). Make-Dateien können folgende Komponenten enthalten:

- Kommentare
- Makros
- Explizite Regeln
- Direktiven
- Implizite Regeln

Symbolische Zielangaben

Eine symbolische Zielangabe veranlaßt MAKE, mehrere Zieldateien mit Hilfe einer einzigen Make-Datei aufzubauen (Sie müssen dabei keine verketteten Abhängigkeiten berücksichtigen). In der Abhängigkeits-Zeile werden alle Zieldateien, die Sie aufbauen wollen, aufgeführt. Sie müssen keine Befehle für ein symbolisches Ziel eingeben.

In der folgenden Make-Datei bewirkt die symbolische Zielangabe *alleDateien*, daß die beiden Dateien DATEI1.EXE und DATEI2.EXE aufgebaut werden.

```
alleDateien: datei1.exe datei2.exe    # Zu beachten: keine Befehle zu
diesem Ziel angeben
datei1.exe: datei1.obj
```

```
  bcc datei1.obj
datei2.exe: datei2.obj
  bcc datei2.obj
```

Regeln für symbolische Ziele

Im Zusammenhang mit symbolischen Zielen sind folgende Regeln zu beachten:

- Symbolische Ziele benötigen keine Befehlszeile.
- Geben Sie jedem symbolischen Ziel einen eigenen, eindeutigen Namen; dies darf nicht der Name einer Datei im aktuellen Verzeichnis sein.
- Wählen Sie für ein symbolisches Ziel einen Namen, der den Betriebssystemkonventionen für Dateinamen entspricht.

Explizite und implizite Regeln

Explizite und implizite Regeln, nach denen MAKE arbeitet, sind grob gesehen wie folgt definiert:

- Explizite Regeln geben MAKE Anweisungen für bestimmte Dateien.
- Implizite Regeln geben MAKE allgemeine Anweisungen für Fälle, in denen keine explizite Regel existiert.

Die Regeln haben folgendes Format:

```
Abhängigkeits-Zeile
  Befehle
  .
  .
  .
```

Die Abhängigkeits-Zeile ist für explizite und implizite Regeln unterschiedlich, aber die Befehle sind dieselben (für Informationen über verkettete Abhängigkeiten siehe den Einleitungsabschnitt *Arbeitsweise von MAKE*).

MAKE erlaubt mehrere Regeln für ein Ziel. Sie können nach der ersten expliziten Regel noch Zeilen mit Dateien, von denen das Ziel abhängt, anfügen, aber nur eine Zeile darf Befehle enthalten. Beispiel:

```
ziel1: Quelle1 quelle2 quelle3 quelle4 quelle5
ziel1: quelle6 quelle7 quelle8
bcc -c $**
```

Syntax für explizite Regeln

Explizite Regeln sind Anweisungen an MAKE, die vollständige Dateinamen beinhalten. In einer expliziten Regel werden ein oder mehrere Zieldateien angegeben, danach folgen ein oder zwei Doppelpunkte. Ein Doppelpunkt bedeutet, daß die Make-Datei für das Ziel nur eine Regel enthält; zwei Doppelpunkte bedeuten, daß zwei oder mehr Regeln für das Ziel vorhanden sind.

Die Syntax für explizite Regeln lautet:

Die geschweiften Klammern sind erforderlich, wenn der Parameter *pfad* angegeben wird.

```
ziel  [ziel...]:[:][{pfad}]  [quelle(n)...]
  befehle
  :
```

- *ziel* — Der Name (mit Erweiterung) der Datei, die aktualisiert werden soll (*ziel* muß am Zeilenanfang stehen, davor sind auch keine Leer- oder Tabulatorzeichen erlaubt). Ein oder mehrere Ziele müssen durch Leer- oder Tabulatorzeichen getrennt werden. Der Name einer Zieldatei darf in einer Make-Datei nicht mehr als einmal an der Position *ziel* einer expliziten Regel angegeben werden.
- *pfad* — Eine Liste von Verzeichnissen (durch Kommata getrennt und in geschweifte Klammern eingeschlossen), in denen die Quellendateien zu finden sind.
- *quellen* — Die Datei(en), deren Erstellungsdatum und -uhrzeit MAKE prüfen soll, um festzustellen, ob sie jünger als *ziel* sind (vor quellen *muß* ein Leerzeichen stehen). Wenn eine Quellendatei in derselben Make-Datei auch als Zieldatei auftritt, so aktualisiert oder erzeugt MAKE sie erst als Zieldatei, bevor sie als Quellendatei für ein anderes Ziel benutzt wird.
- *befehle* — Ein beliebiges Betriebssystemkommando. Eine Regel kann mehrere Befehle enthalten. Die Befehle müssen mit mindestens einem Leer- oder Tabulatorzeichen eingerückt sein (siehe den Abschnitt *Befehlssyntax*).

Wenn eine Abhängigkeits- oder Befehlszeile (zu) lang ist, kann ein umgekehrter Schrägstrich (\) benutzt werden, um MAKE mitzuteilen, daß die Zeile in der nächsten fortgesetzt wird. Setzen Sie den Schrägstrich in der Zeile hinter den Namen einer Ziel- oder Quellendatei. Beispiel:

```
MEINS.EXE:      DATEI1.OBJ\
  DATEI2.OBJ\
  DATEI3.OBJ
    bcc datei1.obj datei2.obj datei3.obj
```

Mehrere Regeln für ein Ziel

Zu einem einzigen Ziel können mehrere explizite Regeln gehören. Um MAKE mitzuteilen, daß mehrere explizite Regeln vorhanden sind, setzen Sie zwei Doppelpunkte (::) hinter den Namen des Ziels. Das folgende Beispiel zeigt ein Ziel mit mehreren Regeln und Befehlen.

```
.cpp.obj:
   bcc -c -ncobj $

.asm.obj:
   tasm /mx $, asmobj\\

meinelib.lib :: f1.obj f2.obj
   echo C-Dateien hinzufügen

   tlib meinelib -+cobj\f1 -+cobj\f2

meinelib.lib :: f3.obj f4.obj
   echo ASM-Dateien hinzufügen
   tlib meinelib -+asmobj\f3 -+asmobj\f4
```

Syntax für implizite Regeln

Eine implizite Regel beginnt mit einer Pfadangabe oder mit einem Punkt (.), sie *impliziert* eine Ziel-Quellendatei-Beziehung. Die Hauptbestandteile der Regel sind Dateinamenserweiterungen, getrennt durch Punkte. Die erste Erweiterung gehört zur Quellen-, die zweite zur Zieldatei.

Wenn eine durch eine implizite Regel bestimmte Quellendatei »veraltet« (verglichen mit der Zieldatei) ist oder nicht existiert, so führt MAKE die in der Regel angegebenen Befehle aus. MAKE aktualisiert in expliziten Regeln angegebene Quellendateien vor den in impliziten Regeln angegebenen.

Die Syntax für implizite Regeln lautet:

```
[{quell_verz}].quell_erw[{ziel_verz}].ziel_erw:
  [befehle]
```

- {*quell_verz*} Das Verzeichnis der Quellendateien. Mehrere Verzeichnisse werden durch Semikolons getrennt.
- .*quell_erw* Die Namenserweiterung der Quellendateien.
- {*ziel_verz*} Das Verzeichnis der Zieldateien (ausführbare Dateien). Mehrere Verzeichnisse werden durch Semikolons getrennt.
- .*ziel_erw* Die Namenserweiterung der Zieldateien. Hier sind auch Makros erlaubt.
- **:** Kennzeichnet das Ende der Abhängigkeits-Zeile.
- *befehle* Ein beliebiges Betriebssystemkommando. Eine Regel kann mehrere Befehle enthalten. Die Befehle müssen mit mindestens einem Leer- oder Tabulatorzeichen eingerückt sein (siehe den Abschnitt *Befehlssyntax*).

Wenn zwei implizite Regeln zu einer Ziel-Namenserweiterung passen, aber keine Quellendatei existiert, so benutzt MAKE diejenige implizite

Regel, in der diejenige Quellendatei-Namenserweiterung steht, die in der **.suffixes**-Liste zuerst erscheint. Siehe auch den Abschnitt über **.suffixes**.

Explizite Regeln mit impliziten Befehlen

Ein Ziel in einer expliziten Regel kann seine Befehlszeile von einer impliziten Regel »ausleihen«. Das folgende Beispiel zeigt eine implizite Regel und eine explizite Regel ohne Befehlszeile.

```
.c.obj:
    bcc -c $<     # Dieser Befehl benutzt das später beschriebene Makro
meinprog.obj:    # Diese explizite Regel verwendet den Befehl bcc -c
meinprog.c.
```

Siehe auch den Abschnitt *Vordefinierte Makros.*

Der Befehl in der impliziten Regel weist MAKE an, MEINPROG.C zu compilieren (das Makro **$<** ersetzt den Namen `meinprog.obj` durch `meinprog.c`).

Befehlssyntax

Als Befehle dürfen alle Betriebssystemkommandos verwendet werden, außerdem aber auch MAKE-Makros, Direktiven und spezielle Operatoren, die das Betriebssystem nicht kennt (Hinweis: Der senkrechte Strich **|** kann in Befehlen nicht benutzt werden). Hierzu im folgenden einige Beispiele:

```
cd..

bcc -c meinprg.c

COPY *.OBJ C:\PROJEKTA

bcc -c $(QUELLE)      # Die Makros werden unten in diesem Kapitel erklärt.
```

Für Befehle gilt folgende allgemeine Syntax:

```
[praefix] befehle
```

Befehls-Präfixe

Sowohl in impliziten als auch in expliziten Regeln dürfen die Befehle mit Präfixen versehen sein, die die Art und Weise verändern, in der MAKE die Befehle behandelt. Tabelle 12.2 zeigt die in Make-Dateien verwendbaren Präfixe. Jedes Präfix wird im Anschluß an die Tabelle noch genauer erklärt.

Tabelle 12.2: Befehls-Präfixe

Präfix	Beschreibung
@	Verhindert, daß MAKE den Befehl bei der Ausführung anzeigt.
-*num*	Stoppt die Abarbeitung der Befehle in der Make-Datei, wenn ein Befehl einen Exit-Code größer als *num* zurückgibt. Im Normalfall bricht MAKE bei jedem Exit-Code ungleich Null ab. Zwischen - und *num* ist kein Whitespace zulässig.
-	Setzt die Abarbeitung der Befehle in der Make-Datei fort, ganz gleich, welchen Exit-Code sie zurückgeben.
&	Erweitert entweder das Makro **$****, das alle Quellendateien repräsentiert, oder das Makro **$?**, das alle Quellendateien, die »jünger« als das Ziel sind, repräsentiert. Der Befehl wird einmal für jede Quellendatei, deren Name durch die Makroerweiterung erzeugt wird, ausgeführt.

Verwendung von @

Der folgende Befehl wird mit dem Präfix **@** verwendet, das verhindert, daß der Befehl bei der Ausführung auf dem Bildschirm angezeigt wird.

```
diff.exe : diff.obj
    @bcc diff.obj
```

Verwendung von num und -

Die Präfixe **num** und - legen fest, wie MAKE die Verarbeitung beim Auftreten von Fehlern fortsetzt. Damit bestimmen Sie, daß MAKE bei jedem Fehler bzw. nur bei Fehlern mit einem maximalen vorgegebenen Exit-Code weiterarbeitet.

Im folgenden Beispiel setzt MAKE die Verarbeitung auch fort, nachdem BCC nicht erfolgreich compiliert hat.

```
ziel.exe : ziel.obj
ziel.obj : ziel.cpp
  bcc -c ziel.cpp
```

Verwenden von &

Das Präfix **&** bewirkt, daß der Befehl auf jede Quellendatei einmal angewendet wird. Er ist besonders nützlich für solche Befehle, die als Parameter keine Liste von Dateien akzeptieren. Im Beispiel

```
copyalle : datei1.cpp datei2.cpp
    &copy $** c:\temp
```

wird COPY zweimal aufgerufen, nämlich so:

```
copy datei1.cpp c:\temp
copy datei2.cpp c:\temp
```

Ohne das Präfix **&** würde COPY nur einmal aufgerufen.

Befehlsoperatoren

Im Befehlsteil einer Make-Datei dürfen Sie jedes Betriebssystemkommando verwenden. MAKE arbeitet mit den normalen Operatoren (wie **+**, **-** usw.), kennt aber zusätzlich auch andere nützliche.

12.3: Kommandozeilen-Operatoren

Operator	Beschreibung
<	Liest die Eingabedaten für den Befehl aus einer Datei statt von der Standardeingabeeinheit.
>	Schreibt die Ausgabedaten des Befehls in eine Datei.
>>	Fügt die Ausgabedaten des Befehls am Ende einer Datei an.
<>	Erzeugt eine temporäre Datei und benutzt ihren Inhalt als Eingabe für den Befehl.
&&	Erzeugt eine temporäre Datei und fügt ihren Namen in die Make-Datei ein.
begrenzer	Ein beliebiges Zeichen außer **#** und ****, das zusammen mit **<** und **&&** als Beginn- und Endezeichen für eine temporäre Datei benutzt wird. Alle Zeichen in derselben Zeile, die dem Beginnzeichen unmittelbar folgen, werden ignoriert. Das Endezeichen muß allein in einer Zeile stehen.

Fehlersuche mit Hilfe von temporären Dateien

Temporäre Dateien können Ihnen bei der Fehlersuche in einem Befehlsblock nützlich sein. Dazu läßt man MAKE die tatsächlich von ihm auszuführenden Befehle in eine temporäre Datei schreiben. Die Namen der temporären Dateien beginnen mit MAKE0000.@@@, die Zahl 0000 wird mit jeder weiteren temporären Datei, die Sie beibehalten lassen, hochgezählt. Nach **&&** und nach den Befehlen, die Sie in die temporäre Datei geschrieben haben möchten, müssen Begrenzerzeichen stehen (ein sinnvolles Begrenzerzeichen ist das Ausrufezeichen **!**).

Das folgende Beispiel zeigt, wie man MAKE mit **&&** dazu bringen kann, eine Datei mit den Eingabedaten für TLINK zu erzeugen.

```
prog.exe: A.obj B.obj
    TLINK /c &&!
    c0s.obj $**
    prog.exe
    prog.map
    maths.lib cs.lib
    !
```

Die dabei mit Hilfe von **&&** erzeugte Response-Datei enthält folgende Befehle:

```
c0s.obj a.obj b.obj
prog.exe
prog.map
maths.lib cs.lib
```

Verwenden von MAKE-Makros

Bei Makros ist die Groß-/Kleinschreibung wichtig. MAKRO1 ist nicht dasselbe wie Makro1.

Ein MAKE-Makro ist ein String, der immer dann erweitert (eingesetzt) wird, wenn das Makro in einer Make-Datei aufgerufen wird. Mit Makros können Sie Schablonen für Make-Dateien anlegen und sie danach für die verschiedenen Projekte anpassen. Um beispielsweise ein Makro zu definieren, das LIBNAME heißt und den String *meinelib.lib* repräsentiert, geben Sie `LIBNAME = meinelib.lib` ein.

Immer wenn MAKE später auf das Makro `$(LIBNAME)` stößt, setzt es dafür den String `meinelib.lib` ein.

Wenn MAKE in der Make-Datei ein nichtdefiniertes Makro findet, sucht es nach einer Umgebungsvariablen des Betriebssystems (normalerweise mit SET definiert), die diesen Namen hat, und setzt deren Definition als Makroerweiterungstext ein. Wenn Sie beispielsweise in der Make-Datei `$(path)` geschrieben, *path* aber nicht als Makro definiert haben, so benutzt MAKE dafür den Text, den Sie für PATH in Ihrer Datei AUTOEXEC.BAT

definiert haben. (Für Informationen über das Definieren von Umgebungsvariablen siehe Ihre Betriebssystemdokumentation.)

Makros definieren

Die allgemeine Syntax für die Definition eines Makros in einer Make-Datei lautet:

```
MakroName = erweiterungs_text
```

- *MakroName* kann aus bis zu 512 Zeichen bestehen, wobei Groß- und Kleinschreibung unterschieden wird.
- *erweiterungs_text* kann aus bis zu 4096 Zeichen (nur Buchstaben, Ziffern, Interpunktionszeichen und Whitespace) bestehen.

Jedes Makro muß in einer eigenen Zeile der Make-Datei stehen. Normalerweise setzt man die Makros an den Anfang der Datei. Wenn MAKE mehr als eine Definition für *MakroName* findet, so ersetzt die neue Definition die alte.

Makros lassen sich auch mit Hilfe der Kommandozeilenoption **-D** (siehe Tabelle 12.1) definieren. Es kann auch mehr als ein Makro definiert werden, die einzelnen Makros werden dabei durch Leerzeichen getrennt. Die folgenden Beispiele zeigen, wie Makros in der Kommandozeile definiert werden:

```
make -Dquellverz=c:\projekta
make command="bcc -c"
make command=bcc option=-c
```

Zwischen der Definition von Makros in der Kommandozeile und in der Make-Datei bestehen folgende Syntaxunterschiede:

Tabelle 12.4: Syntaxunterschiede der Makrodefinition per Kommandozeile und per Make-Datei

Syntax	Make-Datei	Kommandozeile
Leerzeichen vor und nach = erlaubt	Ja	Nein
Leerzeichen vor *MakroName* erlaubt	Nein	Ja

Makros aufrufen

Um ein Makro in einer MAKE-DATEI aufzurufen, geben Sie `$(MakroName)` ein, wobei *MakroName* der Name irgendeines definierten Makros ist. Schließen Sie den Makronamen dabei entweder in geschweifte {} oder in runde () Klammern ein.

MAKE erweitert Makros zu verschiedenen Zeitpunkten, je nachdem, wo sie in der Make-Datei stehen:

- Innere Makros in einer Verschachtelung werden erweitert, wenn das äußere Makro aufgerufen wird.
- Makros in Regeln und Direktiven werden erweitert, wenn MAKE die Make-Datei das erste Mal liest.

- Makros in Befehlen werden erweitert, wenn der Befehl ausgeführt wird.

Ersetzen von Strings in Makros

MAKE ermöglicht es Ihnen, in einem zuvor definierten Makro einige Zeichen zeitweilig durch andere zu ersetzen. Wenn Sie zum Beispiel das Makro *QUELLE* durch `QUELLE = f1.cpp f2.cpp f3.cpp` definiert haben, so können Sie die Zeichen .OBJ durch die Zeichen .CPP ersetzen, indem Sie den Makroaufruf in der Form `$(QUELLE:.CPP=.OBJ)` schreiben. Durch diese Ersetzung wird das Makro nicht neu (anders) definiert.

Regeln für die eben beschriebenen Ersetzungen in Makros:

- Syntax: $(MakroName:original_text=neuer_text).
- Whitespace vor oder hinter dem Doppelpunkt ist unzulässig.
- Der String *original_text* muß exakt (einschließlich Groß-/Kleinschreibung) dem entsprechenden String in der Makrodefinition gleichen.

MAKE erlaubt es auch, Makros als *neuer_text* einzusetzen. Beispiel:

```
MEINEERW=.C
QUELLE=f1.cpp f2.cpp f3.cpp
$(QUELLE:.cpp=$(MEINEERW))  # Bewirkt Ersetzung von f1.cpp durch f1.c usw.
```

Vordefinierte MAKE-Makros

MAKE kennt mehrere vordefinierte Makros, die Sie in Ihren Make-Dateien aufrufen können. Tabelle 12.5 zeigt die Definitionen dieser vordefinierten Makros und außerdem ihre Form nach der Erweiterung innerhalb von expliziten und impliziten Regeln.

Tabelle 12.5: Vordefinierte Makros

Makro	Erweit. in impl. Regel	Erweit. in expl. Regel	Beispiel
$*	Pfad + Datei (Quelle)	Pfad + Datei (Ziel)	C:\PROJEKTA\MEINZIEL
$<	Pfad + Datei + Erw. (Quelle)	Pfad + Datei + Erw. (Ziel)	C:\PROJEKTA\MEINZIEL.OBJ
$:	Pfad (Quellen)	Pfad (Ziel)	C:\PROJEKTA
$.	Datei + Erw. (Quelle)	Datei + Erw. (Ziel)	MEINEQU.C
$&	Datei (Quelle)	Datei (Ziel)	MEINEQU
$@	Pfad + Datei + Erw. (Ziel)	Pfad + Datei + Erw. (Ziel)	C:\PROJEKTA\MEINEQU.C
$**	Pfad + Datei + Erw. (Quelle)	Alle Quellendateien + Erw.	DATEI1.CPP DATEI2.CPP DATEI3.CPP
$?	Pfad + Datei + Erw. (Quelle)	Veraltete Dateien (Quelle)	DATEI1.CPP

Tabelle 12.6: Weitere vordefinierte Makros

Makro	Erweitert zu	Bemerkung
`__MSDOS__`	1	Falls MAKE unter DOS läuft.
`__MAKE__`	0x0370	Versionsnummer von Make (hexadezimal).
`MAKE`	make	Name der ausführbaren Datei, die MAKE enthält.
`MAKEFLAGS`	Optionen	In der Kommandozeile angegebene Optionen.
`MAKEDIR`	Verzeichnis	Verzeichnis, in dem MAKE.EXE liegt.

Verändern vordefinierter Makros

Wenn keines der in Tabelle 12.5 angegebenen Makros bei der Erweiterung genau den String liefert, den Sie wollen, so benutzen Sie zusätzliche Makromodifizierer. Diese dienen dazu, die gewünschten Teile aus dem String »herauszuschneiden«.

Die Syntax zum Modifizieren eines vordefinierten Makros lautet:

```
$(MakroName [Modifizierer])
```

Tabelle 12.7 zeigt die möglichen Makromodifizierer mit je einem Anwendungsbeispiel.

Tabelle 12.7: Makromodifizierer für Dateinamen

Modifizierer	Erweiterung enthält	Beispiel	Ergebnis
D	Laufwerk + Verzeichnis	$(<D)	C:\PROJEKTA\
F	Dateiname + Erweiterung	$(<F)	MEINEQU.C
B	Dateiname allein	$(<B)	MEINEQU
R	Laufwerk, Verzeichnis + Dateiname	$(<R)	C:\PROJEKTA\MEINEQU

Verwenden von MAKE-Direktiven

MAKE-Direktiven ähneln den Direktiven von Sprachen wie C und Pascal, sie ermöglichen viele verschiedene Aktionen, wie zum Beispiel das Anzeigen von Befehlen vor der Ausführung auf dem Bildschirm. Eine MAKE-Direktive beginnt entweder mit einem Ausrufungszeichen oder mit einem Punkt. Tabelle 12.8 zeigt alle MAKE-Direktiven und die korrespondierenden Kommandozeilenoptionen (Direktiven haben Vorrang vor Kommandozeilenoptionen). Jede Direktive wird im Anschluß an die Tabelle genauer erläutert.

Tabelle 12.8: MAKE-Direktiven

Direktive	Option	Beschreibung
.autodepend	**-a**	Aktiviert die automatische Abhängigkeitsprüfung.
!elif		Wirkt wie **else if** bei C.
!else		Wirkt wie **else** bei C.
!endif		Schließt eine **!if**-, **!ifdef**- oder **!ifndef**-Anweisung ab.
!error		Läßt MAKE anhalten und eine Fehlermeldung ausgeben.
!if		Beginn eine Bedingungsanweisung.
!ifdef		Wirkt wie **ifdef** bei C, bezieht sich aber auf Makros statt auf **#define**-Direktiven.
!ifndef		Negation von **!ifdef**
.ignore	**-i**	Veranlaßt MAKE, den Rückgabewert eines Befehls zu ignorieren.
!include		Spezifiziert eine Datei, die in die Make-Datei eingefügt werden soll.

!message		Bewirkt die Ausgabe einer Meldung beim Abarbeiten der Make-Datei.
.noautodepend	**-a-**	Deaktiviert die automatische Abhängigkeitsprüfung.
.noIgnore	**-i-**	Macht **.ignore** rückgängig.
.nosilent	**-s-**	Bewirkt, daß MAKE jeden Befehl vor seiner Ausführung anzeigt.
.noswap	**-S-**	Veranlaßt MAKE, sich nicht selbst aus dem Speicher auszulagern, bevor ein Befehl ausgeführt wird.
.path.ext		Veranlaßt MAKE, Dateien mit der Namenserweiterung *ext* im angegebenen Verzeichnis zu suchen.
.precious		Bewirkt, daß die Zieldatei(en) nicht gelöscht werden, auch wenn bei ihrer Erzeugung ein Fehler auftrat.
.silent	**-s**	Bewirkt, daß die Befehle vor ihrer Ausführung nicht angezeigt werden.
.suffixes		Legt die implizite Regel für Fälle fest, in denen die Angabe der Quelle mehrdeutig ist.
.swap	**-S**	Veranlaßt MAKE, sich selbst aus dem Speicher auszulagern, bevor ein Befehl ausgeführt wird.
!undef		Löscht die Definition eines Makros.

.autodepend

Die automatische Abhängigkeitsprüfung bezieht sich auf OBJ-Dateien, zu denen es korrespondierende CPP-, C- oder ASM-Dateien gibt. Ist die Option **.autodepend** aktiviert, so prüft MAKE die Datums-/Uhrzeitangaben aller Dateien, die an der Erzeugung einer OBJ-Datei beteiligt sind. Wenn eine Datei von den Datums-/Uhrzeitangaben her auf neuerem Stand ist als die davon abhängige OBJ-Datei, so wird durch Neuübersetzung eine aktuelle OBJ-Datei erzeugt. Sie können die Option **.autodepend** bzw. **-a** anstelle der verketteten Abhängigkeitsprüfung (siehe dazu den Einleitungsabschnitt *Arbeitsweise von MAKE*) verwenden.

!error

Die Syntax der Direktive **.error** lautet:

```
!error beliebiger_text
```

Wenn MAKE auf diese Direktive stößt, stoppt es den Ablauf und zeigt den folgenden Text an:

```
Fatal makefile exit code: Error directive: beliebiger_text
```

Die Direktive **!error** ist zum Einsatz in bedingten Verzweigungen vorgesehen; Sie können damit das Anhalten der Verarbeitung und die Ausgabe einer Fehlermeldung erreichen, siehe dazu folgendes Beispiel:

```
!if !$d(MEINMAKRO)
# wenn MEINMAKRO nicht definiert ist:
!error MEINMAKRO ist nicht definiert
!endif
```

Wenn MEINMAKRO beim vorstehenden Beispiel nicht definiert ist, gibt MAKE die folgende Meldung aus:

```
Fatal makefile 4: Error directive: MEINMAKRO ist nicht definiert
```

Alle Optionen zum Abschalten der Fehlerprüfung

Es gibt vier Optionen zum Abschalten der Fehlerprüfung:

- Die Direktive **.ignore** deaktiviert die Fehlerprüfung für einen bestimmten Teil der Make-Datei.
- Die Kommandozeilenoption **-i** deaktiviert die Fehlerprüfung für die gesamte Make-Datei.
- Das Befehlspräfix **-num**, der als Teil einer Regel angegeben wird, beeinflußt die Fehlerprüfung für den zugehörigen Befehl in der Weise, daß nur ein Exit-Code, der größer als die vorgegebene Zahl ist, zum Abbruch führt.
- Das Befehlspräfix **-** deaktiviert die Fehlerprüfung für den zugehörigen Befehl unabhängig vom Exit-Code.

!if und andere Bedingungsdirektiven

Die Direktive **!if** arbeitet ähnlich wie die **if**-Anweisungen der Sprache C (informieren Sie sich über **if**-Anweisungen ggf. im *Borland C++ Programmierhandbuch*). Die Ähnlichkeit von **!if** und den anderen Bedingungsdirektiven mit den Anweisungen für bedingte Compilierung wird an der folgenden Syntax deutlich:

```
!if bedingung    !if bedingung             !if bedingung  !if makro
  .  .            .                         .
  .  .            .                         .
  .  .            .                         .
!endif           !else                     !elif bedingung          !endif
     .            .
     .            .
     .            .
     !endif      !endif
```

Die folgenden Ausdrücke sind jeweils gleichwertig:

`!ifdef makro` und `!if $d(makro)`
`!ifndef makro` und `!if !$d(makro)`

Für die Bedingungsdirektiven gelten folgende Regeln:

- Zwischen den Direktiven **!if**, **!ifdef** oder **!ifndef** einerseits und der zugehörigen Direktive **!endif** andererseits darf nur maximal eine **!else**-Direktive stehen.

- Zwischen den Direktiven **!if**, **!ifdef**, **!ifndef** oder **!else** einerseits und der zugehörigen Direktive **!endif** andererseits dürfen beliebig viele **!elif**-Direktiven stehen.
- Eine Regel darf nicht über Zweige von Bedingungsdirektiven hinweg »verteilt« werden.
- Bedingungsdirektiven dürfen verschachtelt werden.
- Die zu der Direktive **!if**, **!ifdef** oder **!ifndef** gehörende **!endif**-Direktive muß innerhalb derselben MAKE-Quelltextdatei stehen.

Zwischen den Direktiven **!if** und **!endif** darf folgendes stehen:

- Makrodefinitionen
- **!include**-Direktiven
- Explizite Regeln
- **!error**-Direktiven
- Implizite Regeln
- **!undef**-Direktiven

Bedingung in der Syntaxbeschreibung der **!if**- und **!elif**-Direktiven steht für einen Bedingungsausdruck, der aus dezimalen, oktalen oder hexadezimalen Konstanten sowie aus den in Tabelle 12.9 aufgeführten Operatoren besteht.

Tabelle 12.9: Operatoren in Bedingungsausdrücken

Operator	Beschreibung	Operator	Beschreibung
–	Negation	**?:**	Bedingungsausdruck
~	Bitweise Negation	**!**	Logisches NICHT
+	Addition	**>>**	Rechtsschieben
-	Subtraktion	**<<**	Linksschieben
*****	Multiplikation	**&**	Bitweises UND
/	Division	**\|**	Bitweises ODER
%	Rest (Modul)	**^**	Bitweises Exklusiv-Oder
&&	Logisches UND	**>=**	Größer oder gleich*
\|\|	Logisches ODER	**<=**	Kleiner oder gleich*
>	Größer als	**==**	Gleich*
<	Kleiner als	**!=**	Ungleich*

* Operator auch für String-Ausdrücke anwendbar.

MAKE wertet jeden Bedingungsausdruck so aus, daß das Ergebnis entweder eine einfache, vorzeichenbehaftete 32-Bit-Ganzzahl oder ein String ist.

!include

Die Direktive **!include** ähnelt der Präprozessor-Direktive **#include** in den Sprachen C und C++, sie gestattet es Ihnen, in die Make-Datei Text aus einer anderen Datei einzubeziehen:

```
!include dateiname
```

Schließen Sie *dateiname* in Anführungszeichen (**""**) oder in spitze Klammern () ein. Sie dürfen Dateien mit **!include**-Direktiven beliebig tief verschachteln; es ist jedoch unzulässig, dieselbe **!include**-Direktive mehrfach in einer Make-Datei zu verwenden (Fehlermeldung: `cycle in the include file`).

Regeln, Befehle und Direktiven müssen immer innerhalb derselben Quelltextdatei enden, in der sie beginnen; Sie können einen Befehl also nicht in einer Include-Datei beginnen und in der Make-Datei enden lassen.

Wenn Sie nicht mit Hilfe der Option **-I** explizit ein Verzeichnis vorgeben, sucht MAKE die Include-Dateien im aktuellen Verzeichnis.

!message

Die Direktive **!message** dient dazu, während der Abarbeitung einer Make-Datei Meldungen auf dem Bildschirm ausgeben zu lassen. Solche Meldungen können hilfreich sein, wenn eine Make-Datei nicht wie gewünscht arbeitet und Sie den Fehler suchen müssen. Wenn Sie zum Beispiel Probleme mit einer Makrodefinition haben, nehmen Sie folgende Zeile in Ihre Make-Datei auf:

```
!message Das Makro ist hier definiert als: $(MakroName)
```

Wenn MAKE diese Zeile abarbeitet, zeigt es auf dem Bildschirm die Meldung `Das Makro ist hier definiert als: .CPP` an (wenn die Makroerweiterung auf den String .CPP führt). Eine Folge von **!message**-Direktiven hilft Ihnen so, den Fehler in Ihrer Make-Datei zu finden.

.path.ext

Die Direktive **.path.ext** teilt MAKE mit, wo es nach Dateien mit einer bestimmten Namenserweiterung suchen soll. Im folgenden Beispiel wird MAKE angewiesen, nach Dateien mit der Erweiterung C in den Verzeichnissen C:\QUELLEN und C:\CDATEIEN und nach Dateien mit der Erweiterung OBJ im Verzeichnis C:\OBJS zu suchen:

```
.path.c = C:\QUELLEN;C:\CDATEIEN
.path.obj = C:\OBJS
```

.precious

Wenn MAKE eine Datei nicht korrekt und vollständig aufbauen kann, wird sie am Ende gelöscht. Die Direktive **.precious** verhindert das Löschen; erforderlich ist diese Möglichkeit für bestimmte Arten von Zieldateien wie zum Beispiel Bibliotheken. Wenn sich eines der Module nicht zur Bibliothek hinzufügen läßt, wollen Sie vielleicht nicht, daß die Bibliothek gelöscht wird.

Die Syntax für **.precious** lautet:

```
.precious: ziel [ziel] . . . [ziel]
```

.suffixes

Die Direktive **.suffixes** teilt MAKE die bei der Auswertung von impliziten Regeln zu beachtende Reihenfolge (anhand von Dateinamenserweiterungen) mit.

Für die Direktive **.suffixes** gilt die Syntax:

```
.suffixes: .erw [.erw] . . . [.erw]
```

Dabei steht *.erw* für die Namenserweiterung einer Datei, von der andere abhängen. So weist beispielsweise die Direktive `.suffixes: .asm .c .cpp` MAKE an, bei der Auswertung von impliziten Regeln nicht zu berücksichtigen, in welcher Reihenfolge sie in der Make-Datei stehen. Statt dessen soll MAKE mit denjenigen Regeln beginnen, die eine Abhängigkeit von ASM-Dateien beschreiben, danach kommen entsprechend die C-Dateien und zum Schluß die CPP-Dateien.

Das folgende Beispiel zeigt eine Make-Datei mit einer **.suffixes**-Direktive, die MAKE veranlaßt, für den Aufbau des Programms MEINPROG.EXE zuerst nach einer Quelltextdatei mit der Erweiterung ASM, dann nach einer solchen mit der Erweiterung C und schließlich mit der Erweiterung CPP zu suchen. Falls MAKE die Datei MEINPROG.ASM findet, so läßt es daraus von TASM die Datei MEINPROG.OBJ erzeugen und ruft dann TLINK auf. Andernfalls sucht MAKE nach der Datei MEINPROG.C, um daraus MEINPROG.OBJ zu erzeugen, usw.

```
.suffixes: .asm .c .cpp

meinprog.exe: meinprog.obj
tlink meinprog.obj

.cpp.obj:
   bcc -P $
.asm.obj:
   tasm /mx $
.c.obj:
   bcc -P- $
```

!undef

Die Syntax der Direktive **!undef** lautet:

```
!undef MakroName
```

Diese Direktive löscht die Definition des angegebenen Makros (mit dem Namen *MakroName*); eine später folgende Direktive **!ifdef** *MakroName* ergibt dann das Ergebnis *Falsch*.

Makros innerhalb von Direktiven verwenden

Das Makro **$d** kann zusammen mit der Bedingungsdirektive **!if** verwendet werden, um gewisse Aktionen nur dann durchzuführen, falls ein bestimmtes Makro definiert ist. Hinter **$d** muß ein in runde oder geschweifte Klammern gesetzter Makroname stehen, siehe dazu folgendes Beispiel:

```
!if $d(DEBUG)                # Falls DEBUG definiert ist.
bcc -v f1.cpp f2.cpp         # wird mit Debug-Informationen compiliert,
!else                        # andernfalls
bcc -v- f1.cpp f2.cpp        # ohne Debug-Informationen.
!endif
```

Vorsicht!

Benutzen Sie das Makro **$d** nicht, wenn Sie MAKE mit der Option **-N** aufrufen.

Leere Makros

Ist ein Makro nicht definiert, so ergibt die Abfrage per **!ifdef** *MakroName* das Ergebnis *Falsch*, ein leeres Makro dagegen das Ergebnis *Wahr*. Ein leeres Makro ist ein Makro, in dessen Definition rechts vom Gleichheitszeichen (**=) entweder nur Leerzeichen oder gar keine Zeichen stehen. Die folgende Zeile einer Make-Datei definiert beispielsweise ein leeres Makro:**

```
LEERMAKRO =
```

Ein leeres Makro kann auch in der MAKE-Kommandozeile definiert werden, und zwar mit einer der folgenden Angaben:

```
LEERMAKRO=""
```

oder

```
-DLEERMAKRO
```

K A P I T E L 13

Die Grundlagen des Resource Workshop

Dieses Kapitel bietet einen Überblick über den Resource Workshop und die Windows-Ressourcen. Es erklärt, was Ressourcen sind, und stellt die verschiedenen Ressourcen-Typen dar. Es behandelt auch, wie die verschiedenen Arten von Ressourcen-Dateien im Resource Workshop verwendet werden und wie die Dateien in einem Projekt zusammenwirken.

Was Windows-Ressourcen sind

Ressourcen definieren den sichtbaren Teil Ihres Windows-Programms. Wenn Sie zum Beispiel ein Dialogfenster öffnen und einen Schalter drücken, um eine Aufgabe zu erledigen, gehen Sie mit den Ressourcen dieses Programms um. Zusätzlich zu Dialogfenstern und Schaltern können auch andere Ressourcen wie Symbole, Mauszeiger, Bitmaps, Menüs und Tastenkürzel in Ihrer Anwendung verwendet werden.

Im allgemeinen sind die Ressourcen einer Windows-Anwendung getrennt vom Quelltext des Programms, so daß Sie umfangreiche Änderungen an der Benutzeroberfläche vornehmen können, ohne überhaupt den Quelltext des Programms anzusehen.

Da mehrere Anwendungen so denselben Satz von Ressourcen verwenden können, müssen Sie nicht immer dieselben Dialogfenster, Symbole und selbstdefinierten Mauszeiger neu definieren, sondern können diese immer wieder verwenden.

Folgende Typen von Windows-Ressourcen werden unterstützt:

- *Dialogfenster* sind Fenster (gewöhnlich ein Pop-up-Fenster), mit dem Informationen an den Benutzer weitergegeben werden und aus dem der

Benutzer auswählt. Er kann Dateien öffnen, Farben anzeigen, nach Text suichen usw.

- *Menüs* in Windows-Programmen benutzen normalerweise eine Menüleiste, in der die Namen der einzelnen Menüs aufgeführt sind. Ein typisches Menü umfaßt einen oder mehrere Menüeinträge (Befehle). Die meisten Windows-Programme haben zum Beispiel ein Datei-Menü mit Befehlen zum Erstellen, Öffnen, Speichern oder Drucken von Dateien.
- *Tastenkürzel* sind Tastenkombinationen (oder Hot Keys), mit denen der Benutzer in einer Anwendung bestimmte Aufgaben ausführen lassen kann. In einem Windows-Programm kann zum Beispiel das Tastenkürzel [*Umschalt*]+[*Einfg*] für den Befehl Einfügen verwendet werden, mit dem der Benutzer Text oder Grafiken aus der Zwischenablage in eine Datei einfügen kann, die seine Anwendung gerade geöffnet hat. Tastenkürzel erscheinen normalerweise auch im Menü rechts neben den Befehlen, mit denen sie zusammenhängen.
- *String-Tabellen* enthalten Text (wie Beschreibungen, Eingabeaufforderungen und Fehlermeldungen). Der Text wird als Teil des Windows-Programms verwendet. Da diese Text-Strings Windows-Ressourcen sind, die unabhängig vom Programm gespeichert sind, anstatt im Quelltext verstreut zu sein, können Sie oder andere die angezeigten Meldungen eines Programms leicht bearbeiten oder übersetzen (ohne den Quelltext des Programms zu verändern).
- *Bitmaps* sind eine Darstellung von Grafiken im Binärformat. So sind zum Beispiel die verschiedenen Dialogelemente eines typischen Windows-Fensters wie die Pfeiltasten einer Bildlaufleiste, das Symbol des System-Menüs und die Taste zum Verkleinern des Fensters auf Symbolgröße alle als Bitmaps definiert. Jedes Bit, oder jede Bitgruppe, repräsentiert einen Pixel der Grafik.
- *Symbole* sind kleine Bitmaps in der Größe 64x64 Pixel (unterstützt von Windows 3.1 auf Anzeigegeräten mit hoher Auflösung), 32x32 Pixel oder 16x32 Pixel und stellen verkleinerte Fenster dar. Symbole werden mit dem Paint-Editor erzeugt (siehe Kapitel 19).
- *Mauszeiger* sind ebenfalls kleine Bitmaps in der Größe 32x32 Pixel, mit denen die Position der Maus auf dem Bildschirm angezeigt wird. Windows-Programme benutzen selbstdefinierte Mauszeiger, um anzuzeigen, welche Aufgabe der Benutzer gerade bearbeitet. Mauszeiger werden auch mit dem Paint-Editor erzeugt (siehe Kapitel 19).
 Sie können eine Auswahl selbstdefinierter Mauszeiger im Resource Workshop Paint-Editor sehen, wenn Sie eine Bitmap-Ressource bearbeiten (Bitmap, Symbol oder Schrift). Jedesmal, wenn Sie ein neues Zeichenwerkzeug markieren und den Mauszeiger zu der von Ihnen zu bearbeitenden Grafik bewegen, erscheint der Mauszeiger in einer Form, die die auszuführende Aufgabe repräsentiert.

- *Schriften* bezieht sich normalerweise auf eine Menge von Buchstaben und Sonderzeichen. Im Resource Workshop wird unter Schriften allgemeiner ein Satz von Bitmaps verstanden, die zusammen gespeichert und benutzt werden.

 Windows-Programme benutzen Schriften, um das Aussehen, die Größe und den Stil eines Textes festzulegen. Ein Programm kann zum Beispiel auf dem Bildschirm oder Drucker einen Text in **10 Punkt Times Roman Fett** ausgeben. In diesem Fall ist das *Aussehen* durch Times Roman, die *Größe* durch 10 Punkt und der Stil durch *Fett* gegeben.

 Mit dem Resource Workshop können Sie das Aussehen bestehender Schriften verändern oder eigene Schriften erzeugen.

- *Benutzerdefinierte und rcdata-Ressourcen* sind im Resource Workshop eigentlich dasselbe. Sie bestehen aus jeglichen Daten, die Sie in die ausführbare Datei einfügen wollen. So könnten Sie zum Beispiel eine große, nicht zu verändernde Textdatei in Ihre ausführbare Programmdatei als benutzerdefinierte Ressource einfügen.

 Ein häufiger Grund, warum benutzerdefinierte Ressourcen einer Anwendung hinzugefügt werden, ist die verbesserte Speicherverwaltung unter Windows. Viele Windows-Anwendung verwenden das Speichermodell *Medium*, das nur ein einziges Datensegment definiert. Wenn Sie größere Datenmengen verarbeiten und nicht möchten, daß diese während der Laufzeit ständig im Arbeitsspeicher vorhanden sind, können Sie die Daten als verschiebbare benutzerdefinierte Ressource definieren. Eine solche Ressource verbraucht Speicherplatz nur dann, wenn sie gebraucht wird.

- *Version Info*. Resource Workshop unterstützt die VERSIONINFO-Ressource, die Informationen über die ausführbaren Programmdateien (EXE-Dateien) von Windows 3.1 bietet. Eine detaillierte Beschreibung von VERSIONINFO finden Sie in den Online-Hilfe-Dateien von Resource Workshop, oder sehen Sie in Ihrer Dokumentation zu Windows 3.1 nach.

Für die meisten Ressourcen-Typen haben Sie die Wahl: Sie können entweder den leicht zu benutzenden grafischen Ressourcen-Editor verwenden oder mit Hilfe eines Texteditors Ressourcen-Scripts verfassen. So ist Ihnen die Windows-Programmierung auf allen Ebenen zugänglich.

Sie haben zusätzlich zum Grafik- und zum Texteditor des Resource Workshop die Möglichkeit, einen externen Editor zum Bearbeiten von Projektdateien anzugeben. Die Projektdateien werden dann aber beim Öffnen nicht compiliert.

Ressourcen-Dateitypen

Eine mit Resource Workshop erstellte und bearbeitete Datei kann entweder im Binär- oder Textformat vorliegen. Zusätzlich können Standard-Windows-Dateiformate erzeugt werden. Sie können den Resource Workshop daher mit Programmen benutzen, die Binärcode aus Ressourcen-Script-Dateien erzeugen.

- *Ressourcen-Compiler-Script-Dateien*. Eine Ressourcen-Compiler-Datei (.RC) ist eine Textdatei, die Definitionen für eine oder mehrere Ressourcen enthält. Die Datei kann die Definitionen von Ressourcen als Script oder Verweise auf andere Dateien enthalten.
 Sie sollten alle Ihre Resource-Workshop-Projekte auf mindestens einer .RC-Datei aufbauen.
- *Compilierte Ressourcen-Dateien* sind Dateien (.RES), die eine oder mehrere übersetzte Ressourcen enthalten.

 Normalerweise compilieren Sie beim Erstellen eines Windows-Programms alle Ressourcen für Ihre Anwendung in eine einzige .RES-Datei und binden die .RES-Datei dann während des Linkens zu der ausführbaren Datei hinzu. Mit Resource Workshop muß aber keine .RES-Datei erzeugt werden, wenn Sie diese nicht benötigen, da Resource Workshop Ressourcen-Dateien direkt compilieren und an die ausführbare Datei binden kann.
- *Ausführbare Dateien und .DLL-Dateien* (.EXE- und .DLL-Dateien) sind letztendlich die Zieldateien für alle Ihre Ressourcen, die Sie mit dem Resource Workshop definieren. Gewöhnlich compilieren Sie eine .RC-Datei zu einer .RES-Datei und benutzen dann Ihren Compiler, um die .RES-Datei Ihrer ausführbaren oder .DLL-Datei hinzuzufügen. Sie können die Ressourcen auch mit dem Resource Workshop direkt in ausführbare oder .DLL-Dateien einbinden und so den Microsoft Ressourcen Compiler ganz umgehen.

.DRV-Dateien sind Windows-Gerätetreiber und ein Spezialfall der .DLL-Dateien. Sie können die Ressourcen einer .DRV-Datei genauso bearbeiten, wie Sie es auch bei einer .DLL-Datei tun würden.

Wenn Sie die Ressourcen in einer compilierten Binärdatei (einer ausführbaren Datei, einer .DLL-Datei oder einer .RES-Datei) ändern möchten, decompiliert Resource Workshop die Datei, damit Sie sie bearbeiten können und schreibt die darin enthaltenen Ressourcen in die ursprüngliche Datei zurück.

- *Dialogdateien* (.DLG) sind gewöhnlich Ressourcen-Script-Dateien (Textdateien), die Beschreibungen für ein oder mehrere Dialogfenster enthalten. Es gibt aber keine Vorschrift, daß eine .DLG-Datei nur Dia-

logfenster Ressourcen enthalten darf. Es können - wie in einer .RC-Datei - beliebige Ressourcen gespeichert werden.

Bitmap-Ressourcen-Dateien

Es gibt vier Typen von Bitmap-Ressourcen-Dateien:

- Eine *Bitmap*-Datei (.BMP) enthält eine Bitmap-Ressource im Binärformat.
- Eine *Symbol*-Datei (.ICO) enthält ein Symbol im Binärformat.
- Eine *Mauszeiger*-Datei (.CUR) enthält einen benutzerdefinierten Mauszeiger im Binärformat.
- *Schrift*dateien können in zweierlei Form auftreten, als Binärdateien und als Schriftbibliotheken (Font Library).

Eine Schrift-Binärdatei (.FNT) beschreibt eine benutzerdefinierte Schrift im Binärformat. Sie können mit dem Paint-Editor von Resource Workshop eine Schrift entwerfen und in einer .FNT-Datei speichern.

Eine Schrift-Bibliotheksdatei (.FON) .FON-Datei ist eine DLL (Dynamic Link Library) nur für Ressourcen. Die DLL enthält ein Schriftenverzeichnis und eine oder mehrere Schriften. Schriftdateien müssen Sie außerhalb des Resource Workshop erstellen. Sie können aber den Resource Workshop benutzen, um eine bestehende .FON-Datei zu bearbeiten.

T E I L III

Verwenden des Resource Workshop

Teil III dieses Handbuchs erläutert das Erzeugen, Compilieren und Linken von Ressourcen für Anwendungen unter Windows 3.0 und späteren Versionen. Hier erfahren Sie mehr über

- das Arbeiten mit Ressourcen im Text- oder Binärformat
- das Verwalten von Hunderten von Ressourcen in Dutzenden von Dateien
- das Verwenden von Undo- und Redo-Funktionen mit mehreren Ebenen, um erfolgte Änderungen zurückverfolgen zu können
- das Compilieren von Ressourcen nur dann, wenn man sie benötigt
- das Ändern von Ressourcen eines Programms selbst ohne Zugriffsmöglichkeit auf den Quelltext
- das automatische Suchen nach Fehlern wie inkorrekter Syntax oder doppelten Ressourcen-IDs

Teil III umfaßt folgende Kapitel:

- Kapitel 13 führt in die Grundlagen des Resource Workshop ein.
- Kapitel 14 beschreibt Ressourcen und zeigt, wie man sie unter Verwendung von Ressourcenskript-, Binär- und Grafikdateien erzeugt und bearbeitet.
- Kapitel 15 beschreibt das Erzeugen und Bearbeiten von Dialogfeldern.
- Kapitel 16 beschreibt das Erzeugen und Bearbeiten von Menüs.
- Kapitel 17 beschreibt das Erzeugen und Bearbeiten von Tastenkürzeln.
- Kapitel 18 beschreibt das Arbeiten mit String-Tabellen.
- Kapitel 19 beschreibt das Arbeiten mit dem Paint-Editor.

- Kapitel 20 beschreibt das Erzeugen und Bearbeiten von Symbolen.
- Kapitel 21 beschreibt das Erzeugen und Bearbeiten von Mauszeigern.
- Kapitel 22 beschreibt das Erzeugen und Bearbeiten von Schriftressourcen.
- Kapitel 23 beschreibt das Erzeugen und Bearbeiten von benutzerdefinierten Ressourcen.

KAPITEL 14

Arbeiten mit Projekten, Ressourcen und Bezeichnern

Dieses Kapitel beschreibt das Projektfenster und erklärt, wie die Anzeige der Ressourcen geändert und wie Ressourcen dieses Fensters für das Bearbeiten ausgewählt werden können.

Ein *Projekt* ist eine Ansammlung von einer oder mehreren Ressourcen. Eine Projektdatei (normalerweise mit Erweiterung RC) enthält entweder eine oder mehrere Ressourcen oder Referenzdateien, die Ressourcen umfassen, oder beides.

Ein neues Projekt erzeugen

Wenn Sie ein Projekt geöffnet haben, schließt Resource Workshop es jetzt. Sind ungespeicherte Änderungen erfolgt, fragt Resource Workshop, ob Sie die Änderungen speichern wollen, bevor das Projekt geschlossen wird.

Um ein neues Projekt zu erzeugen

1. Wählen Sie *Datei/Neues Projekt*.

 Entscheiden Sie sich, welcher Dateityp die Grundlage Ihres Projekts bilden soll. Üblicherweise ist die Grundlage eines Projekts eine .RC Datei, weil bei diesem Dateityp mit allen Ressourcen gearbeitet werden kann. Prinzipiell können Sie eine Datei des folgenden Typs verwenden.

 - .RC erzeugt eine Ressourcen-Script-Datei.
 - .RES erzeugt eine binäre Ressourcendatei.
 - .CUR erzeugt ein Projekt, was einen Mauszeiger umfaßt.
 - .ICO erzeugt ein Projekt, was ein Symbol umfaßt.
 - .BMP erzeugt ein Projekt, was eine Bitmap umfaßt.
 - .FNT erzeugt ein Projekt, was nur Schrift umfaßt.

2. Klicken Sie auf den Projekt Dateityp, den Sie verwenden wollen. Klicken Sie dann auf OK.

 Resource Workshop zeigt das neue, unbenannte Projekt im Projektfenster an. Für .RC Projekte werden Sie eine Standard Header-Datei angeben müssen, in der Resource Workshop Bezeichner speichert.

Öffnen eines vorhandenen Projekts

1. Wählen Sie *Datei/Projekt öffnen*. Resource Workshop zeigt das Projekt öffnen Dialogfenster.
2. Geben Sie die zu öffnende Datei an, indem Sie wie folgt vorgehen:
 - Geben Sie den Pfad und Dateinamen ein und drücken Sie die [*Eingabetaste*].
 - Wählen Sie einen Dateityp, dann wählen Sie eine Datei aus der Dateiliste und klicken auf *OK*.

 Was Resource Workshop daraufhin macht, hängt davon ab, ob das Projekt eine Binärdatei oder eine Datei mit Ressourcendaten ist.

 - Wenn das Projekt eine binäre Datei, das heißt eine .RES oder .DLL Datei ist, werden die Ressourcen rückübersetzt. Der Fortschritt der Decompilierung wird auf der linken Seite der Statuszeile an dem unteren Ende der Anzeige.
 - Ist das Projekt eine .RC Datei bzw. eine Datei, die Ressourcendaten enthält (wie es normalerweise der Fall ist), liest Resource Workshop direkt die Datei ein (da eine Decompilierung nicht notwendig ist), danach compiliert es die Datei und jede einzelne Resource. Der Fortschritt der Compilierung wird im Status der Compilierung Fenster angezeigt.

 Findet der Compiler einen Fehler, erzeugt Resource Workshop das Compilierungsfehler Dialogfenster, welches den Fehler anzeigt und die Zeile hervorhebt, in der der Fehler gefunden wurde.

 Um zum Bearbeiten der Datei den externen Editor zu verwenden, der in *Datei/Vorgaben* angegeben wurde, klicken Sie auf die Editor starten Taste im Compilierungsfehler Dialogfenster. Wenn Ihr Editor erscheint, machen Sie die notwendigen Verbesserungen, speichern die Datei und verlassen den Editor. Danach muß das Projekt neu geladen werden.
3. Ist das Projekt erst compiliert oder decompiliert, zeigt der Resource Workshop ein Projektfenster, in dem alle Ressourcen aufgelistet sind.

Die Verwendung des Projektfensters

Wenn Sie ein neues oder vorhandenes Projekt geöffnet haben, erzeugt Resource Workshop das Projektfenster. Das Projektfenster dient als Verwaltungswerkzeug für die Datei. Mit dem Projektfenster ist es einfach, einen Gesamtüberblick über das Projekt zu bekommen. Selbst wenn das Projekt eine große Anzahl von Ressourcen enthält, können Sie sie schnell durchsehen, indem Sie im Projektfenster blättern.

Bei einem neuen Projekt ist das Fenster leer und Sie müssen die Ressourcen einfügen, indem Sie sie erzeugen oder als Datei zufügen (später mehr zu diesem Thema).

Bei einem bestehenden Projekt sehen Sie:

- die Liste aller Dateien des Projekts,
- die Ressourcentypen in jeder Datei,
- Wenn die Datei Ressourcendaten enthält (und somit keine .EXE, .RES oder .DLL Datei ist), werden die Bezeichner (**#define** oder Konstanten) angezeigt, die auf Ressourcen verweisen.

Eingebundene und referenzierte Ressourcen

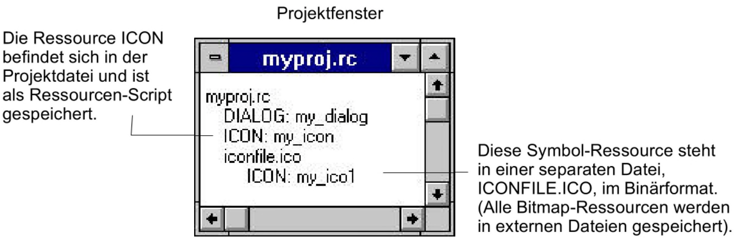
Die Ressourcen Ihres Projekts können entweder in die Datei *eingebunden*, oder es existiert ein Verweis auf die Ressourcen (*referenzierte* Ressourcen)

- Eine eingebundene Ressource wird in der Projektdatei in Form eines Ressourcen-Scripts gespeichert. Sie exisiert nur als Teil des Projekts, in dem sie gespeichert ist, und kann in anderen Projekten nicht benutzt werden.
- Eine referenzierte Ressource ist eine eigene Datei, auf die in der Projektdatei verwiesen wird. Referenzierte Ressourcen mit Dateierweiterungen wie .RC oder .DLG oder Dateien im Binärformat, können auch in anderen Projekten Verwendung finden.

Informationen im Projektfenster anzeigen

Mit dem Menü *Ansicht* wird bestimmt, wie die Informationen im Projektfenster angezeigt werden. Als Standardvoreinstellung zeigt das Projektfenster die ausgewählte Ressource im Fenster an. Mit *Ansicht/Vorschau schließen* können Sie die Anzeige abstellen. Mit *Ansicht/Vertikale Vorschau anzeigen* oder *Ansicht/Horizontale Vorschau anzeigen* können Sie die Anzeige relativ zum Projektfensterrahmen verschieben.

Abbildung 14.1: Anzeigen der Bezeichner nach Datei geordnet



Bezeichner anzeigen

Zeigt alle Bezeichner (**#define** oder Konstanten) im Projekt an.

Ressourcen anzeigen

Listet die Typen und die Namen der Ressourcen auf, wie BITMAP:Flugzeug. Meistens wird diese Option markiert gelassen.

Heben Sie die Markierung an der Option Ressourcen anzeigen auf, wenn Sie sich nur die Dateinamen ansehen wollen, ohne die Ressourcen in den Dateien angezeigt zu bekommen, oder wenn Sie nur an Bezeichnern interessiert sind. Wenn Ressourcen anzeigen abgeschaltet ist, kann keine in der Projektdatei definierte Ressource bearbeitet werden.

Einträge anzeigen

Damit wird wieder ein andere Anzeigemöglichkeit im Projektfenster gewählt. Einträge anzeigen eingeschaltet ist, zeigt Resource Workshop die Einträge innerhalb der einzelnen Ressourcen an (z.B. Popup oder Menüelemente, die in einer Menü-Ressource definiert sind).

Nicht verwendete Typen anzeigen

Listet alle Ressourcentypen auf, unabhängig davon, ob sie im Projekt benutzt worden sind oder nicht.

Einträge anzeigen ist nur verfügbar, wenn *Ansicht/Als Typ* ausgewählt wurde.

Auswählen einer Ressource

Um eine Ressource auszuwählen, benutzen Sie die Maus oder die Pfeiltasten, um die Ressource im Projektfenster hervorzuheben.

- Wenn Sie *Ansicht/Als Typ* ausgewählt haben, suchen Sie zuerst nach dem Ressourcentyp. Die Ressourcen werden nach Namen unter Ihrem Ressourcentyp aufgelistet.
- Wenn Sie *Ansicht/Als Datei* ausgewählt haben, suchen Sie nach der Ressource unter Ihrem Dateinamen, sofern Sie ihn kennen. Dem Namen der Ressource wird der Ressourcentyp und ein Doppelpunkt vorangestellt.

Zum Beispiel wird im Projektfenster myproj.rc (Abbildung 14.1) die Symbol Ressource my_ico1 unter der Datei iconfile.ico in der Form ICON:my_ico1 aufgelistet.

Arbeiten mit Ressourcen

Dieser Abschnitt beschreibt, wie typische Aufgabenstellungen bei der Arbeit mit Ressourcen gelöst werden.

Ressourcen öffnen

Um eine Ressource zu benennen oder zu bearbeiten, muß sie geöffnet werden. Eine der folgenden Vorgehensweisen ist dabei möglich:

- Doppelklicken Sie auf den Ressourcen-Namen (nicht auf den Dateityp) im Projektfenster. Resource Workshop ruft automatisch den geeigneten grafischen Editor auf, wenn einer verfügbar ist. Wenn kein Editor für die Ressource verfügbar ist (z.B. bei einer benutzerdefinierten Ressource), ruft Resource Workshop den internen Texteditor auf.
- Markieren Sie den Ressourcen-Namen im Projektfenster. Wählen Sie dann entweder *Ressource/Bearbeiten* (um einen graphischen Editor zu aufzurufen) oder *Ressource/Als Text* bearbeiten (um den Texteditor zu aufzurufen).

Ressourcen-editoren

Doppelklicken Sie auf einen der folgenden Ressourcen-Typen im Projektfenster (oder markieren Sie den Ressourcen-Typ und wählen *Ressource/Bearbeiten*), so lädt Resource Workshop den dazugehörigen Ressourceneditor:

Die einzelnen Ressourceneditoren werden zusammen mit den Ressourcen in den entprechenden Kapiteln erläutert

- Wenn die Ressource ein Dialogfenster, ein Menü, ein Tastenkürzel oder eine Stringtabelle ist, ruft Resource Workshop den Editor auf, der speziell für die Arbeit mit dieser Ressource gestaltet wurde. Beschreibungen dieser Editoren finden sich in den Kapiteln 15 bis 18.
- Wenn die Resource eine Bitmap-Ressource ist (Symbol, Bitmap, Mauszeiger oder Schrift, zeigt Resource Workshop die Ressource im Paint-Editor an.

Der interne Texteditor

Benutzerdefinierte Ressourcen und zwei vordefinierte Ressourcentypen, nämlich rcdata und VERSIONINFO, können nur mit dem internen Texteditor bearbeitet werden.

Eine Resssource im Binärformat (wie ein Dialogfenster oder ein Menü) wird von Resource Workshop decompiliert, damit Sie mit dem Ressourcenskript arbeiten können.

Ressourcen-Script-Befehle sind in der Online-Hilfe beschrieben.

Der interne Text-Editor ist dem Windows-Notizblock (Editor) sehr ähnlich. Die Tasten [*Entf*], [*Pos1*], [*Ende*], [*Nach oben*], [*Nach unten*], [*Tab*] und [*Rück*] haben dieselbe Wirkung, aber der Editor ist immer im Eingabemodus.

Geben Sie den Text ohne zu formatieren ein, da Resource Workshop den Text beim Compilieren wahrscheinlich umformatiert.

Entdeckt Resource Workshop Fehler, wird unter Angabe der Fundstelle der Text-Editor erneut aufgerufen, do daß Sie den Fehler beheben können.

Sie können auch einen Editor Ihrer Wahl benutzen, um das Ressourcen-Script ohne Zuhilfenahme von Resource Workshop zu bearbeiten. Beachten Sie aber folgendes:

Wenn Resource Workshop die Ressource aufruft, wird die Datei compiliert und das Ressourcen-Script umformatiert. Findet Resource Workshop Kommentare im Text, wenn Sie die Ressource zum Bearbeiten öffnen, wird ein Dialogfenster mit der Warnung angezeigt, daß alle Kommentare gelöscht werden.

Ressourcen einfügen

Sie können eine Ressource direkt in das Projekt einbinden (eingebundene Ressource) oder einen Verweis auf die Datei ins Projekt einfügen (referenzierte Ressource).

Direktes Einbinden einer Ressource

Um eine Ressource einzubinden, öffnen Sie das Projekt, mit dem Sie arbeiten wollen und

1. Wählen Sie *Ressource/Neu* um sich das Neue Ressource Dialogfenster anzeigen zu lassen.
2. Wählen Sie die Header-Datei, in der die Ressource und der Bezeichner erfaßt sein sollen. Beachten Sie, daß Sie nur Header-Dateien auswählen können, die zu Ihrem Projekt gehören. (Benutzen Sie *Datei/Dem Projekt hinzufügen*, um Ihrem Projekt eine neue Header-Datei hinzuzufügen).
3. Doppelklicken Sie auf den Typ der Ressource, die Sie erzeugen wollen.

- Wenn Sie Tastenkürzel, Menü, Dialogfenster oder Stringtabellen erzeugen, fügt Resource Workshop den Eintrag für die Ressource ins Projektfenster ein und ruft den zugehörigen Ressourcen-Editor auf.
- Wenn Sie rcdata oder VERSIONINFO Ressourcen erzeugen, fügt Resource Workshop den Eintrag für die Ressource ins Projektfenster und ruft den internen Editor auf.
- Wenn Sie eine Bitmap-Ressource wie Symbol, Mauszeiger, Bitmap oder Schrift erzeugen, fragt Resource Workshop, ob Sie die Ressource als

- Quelle oder im Binärformat speichern wollen. Wählen Sie die Quelle Option, um die Ressource einzubinden.
- Wenn Sie eine Ressource von bisher nicht aufgeführtem Ressourcen-Typ erzeugen wollen, drücken Sie die Taste *Neuer Typ* und erzeugen Sie einen benutzerdefinierten Ressourcen-Typ für Ihre Ressource. Mehr Informationen über benutzerdefinierte Ressourcen finden Sie in Kapitel 23.

Einfügen einer referenzierten Ressource

Sie können auch eine Ressource durch Dateiverweis einfügen (referenzierte Ressource). Um eine Ressource durch Dateiverweis einzufügen,

1. Wählen Sie *Datei/Dem Projekt hinzufügen*. Resource Workshop zeigt das Dialogfenster »Datei einem Projekt hinzufügen«.
2. Entweder geben Sie den Namen der Ressourcendatei im Dateiname Eingabefeld an, oder Sie doppelklicken auf den Namen der Datei, wenn er im Dateien Listenfenster sichtbar wird.

Durch Auswählen eines Typs in der Dateityp-Liste können Sie Suchkriterien eingeben.

- Wenn die Datei nicht im aktuellen Verzeichnis ist oder einen anderer Dateityp als der aktuelle gewünscht wird, können Sie die Datei auf zweierlei Art auswählen:

Sie können den vollständigen Pfad und Dateinamen in das Dateinamen-Eingabefeld eingeben.

- Sie können das Verzeichnis mit Hilfe des Verzeichnislistenfeldes wechseln. Geben Sie dann ein Suchkriterium für die Datei ins Eingabefeld ein (benutzen Sie z.B. *.CUR, um alle Mauszeiger Ressourcendateien zu finden). Drücken Sie die [*Eingabetaste*]. Sie können auch einen Dateityp aus der Dateityp-Liste auswählen. Wenn der gewünschte Dateiname in der Dateinamenliste erscheint, doppelklicken Sie darauf, um die Datei zu markieren.

3. In der zweiten Auswahlliste (unter dem Feld zum Dateityp) sehen Sie das aktuelle Projekt aufgeführt, in das Sie normalerweise den Verweis auf die Ressourcendatei einfügen. Wenn Ihr Projekt mehr als eine .RC Datei verwendet und Sie den Verweis woanders einfügen wollen, suchen Sie im Listenfenster, bis Sie den Namen der Datei finden, in die Sie den Verweis einfügen wollen.
4. Drücken Sie die [*Eingabetaste*] oder klicken Sie auf *OK*, um die Datei dem Projekt hinzuzufügen. Resource Workshop fügt einen Verweis auf die Datei ins Projektfenster ein.

Bei Auswahl von *Ansicht/Als Datei* werden Sie diesen Dateinamen und die enthaltenen Ressourcen nach Namen aufgelistet sehen. Jede Veränderung, die Sie an einer Ressource im Projekt machen, findet auch in der Datei statt.

So fügen Sie eine neue Ressource mit Dateiverweis ein:

1. Geben Sie den Namen der zu erzeugenden Datei in das Dateinamenfeld des Datei einem Projekt hinzufügen Fensters ein. Wenn Sie andere als die Standard Dateinnamenserweiterungen (wie z.B. .CUR) benutzen wollen, verwenden Sie das Dateitypfeld, um den Typ der zu erzeugenden Ressource zu bestimmen.
2. Klicken Sie auf *OK*. Ein Dialogfenster informiert Sie, daß die Datei noch nicht existiert und fragt Sie, ob Sie die Datei jetzt erzeugen wollen.
3. Bei Klicken auf *OK* erzeugen Sie die Datei. Resource Workshop fügt einen Verweis auf die Datei in das Projektfenster ein.

Verschieben einer Ressource

Sie können mit *Ressource/Umbenennen* eine Ressource von einer Datei des Projekts in eine andere Datei desselben Projekts verschieben. Beide Dateien müssen Teil desselben Projekts sein.

1. Markieren Sie eine Resource, indem Sie sie bearbeiten oder in der Liste der Ressourcen des Projekts markieren.
2. Wählen Sie *Ressource/Umbenennen*. Ein Dialogfenster zeigt Ihnen den Namen der Ressource an sowie die mit Alter Name ausgezeichnete Datei, in der diese Ressource gespeichert ist.
3. Markieren Sie die neue Datei, die die Resource aufnehmen soll, und klicken Sie auf OK.

Ressourcen zwischen Projekten kopieren

Es gibt zwei Möglichkeiten, Ressourcen vom aktuellen Projekt in ein anderes Projekt zu kopieren:

- Eine Möglichkeit ist, die Ressource in einer Datei zu speichern, das aktuelle Projekt zu schließen, ein anderes Projekt zu öffnen und die Ressource als Datei dem neuen Projekt hinzuzufügen. Wenn in der Ressource Bezeichner verwendet werden, müssen Sie aufpassen, daß sie erhalten bleiben, nachdem Sie die Ressource dem neuen Projekt angefügt haben.
- Ein einfacherer Weg ist es, mit zwei Instanzen von Resource Workshop zu arbeiten, wobei jede Instanz ein Projekt geladen hat. Kopieren Sie, wie unter Windows üblich, die Ressource mit Hilfe der Zwischenablage von einem Projekt in ein anderes. Diese Methode ist nicht nur einfacher als die erste Methode, sondern es bleiben auch alle Bezeichner erhalten.

So kopieren Sie eine Ressource mit der zweiten Methode:

1. Öffnen Sie zwei Instanzen von Resource Workshop. Eine umfaßt das Projekt, von dem Sie die Ressource kopieren wollen (Quellprojekt).

Die andere umfaßt das Projekt, in das Sie die Ressource kopieren wollen (Zielprojekt).

2. Stellen Sie sicher, daß dem Zielprojekt eine Bezeichnerdatei beigefügt ist, die die Bezeichner der neuen Ressource aufnehmen kann. (Wenn notwendig, fügen Sie die benötigte Datei mit *Datei/Dem Projekt hinzufügen* ein).

Achtung: [*Strg*]+[*Einfg*] kann Ressourcen nicht korrekt kopieren.

3. Markieren Sie die Ressource, die Sie kopieren wollen, im Projektfenster des Quellprojekts. Kopieren Sie die Ressource mit *Bearbeiten/Kopieren* in die Zwischenablage.
4. Aktivieren Sie das Projektfenster des Zielprojekts und führen Sie *Bearbeiten/Einfügen* aus. Die Resssource wird in das Projekt eingefügt, und das Ressource einfügen Dialogfenster erscheint.
5. Das Ressource einfügen in Feld sollte den Namen des Zielprojekts und den Namen der Bezeichnerdatei enthalten, welche die Bezeichner der zu kopierenden Ressource aufnimmt. Gegebenfalls suchen Sie im Listenfeld, bis der Name der Bezeichnerdatei erscheint, und markieren Sie ihn sodann. Danach drücken Sie Eingabe oder klicken auf *OK*, um die Ressource einzufügen.

Ressourcen löschen

Um eine Ressource aus dem Projekt zu entfernen, markieren Sie die Ressource im Projektfenster und wählen dann *Bearbeiten/Ausschneiden* oder *Bearbeiten/Löschen*, um sie zu löschen. (*Bearbeiten/Ausschneiden* erlaubt Ihnen, die Ressource woanders einzufügen)

Ressourcen umbenennen

So benennen Sie eine Ressource um:

1. Wählen Sie *Ressource/Umbenennen* und das Dialogfenster Ressource umbenennen erscheint.
2. Geben Sie in das Eingabefeld Neuer Name den neuen Ressourcennamen ein und drücken Sie die [*Eingabetaste*].
3. Resource Workshop fragt, ob ein neuer Bezeichner dieses Namens erzeugt werden soll.
 - Wenn Sie *Ja* anklicken, wird die Ressource umbenannt und ihr ein neuer Bezeichnerwert zugewiesen, den Sie bestimmen können.
 - Wenn Sie *Nein* anklicken, wird die Ressource umbenannt, ohne daß eine neuer Bezeichner erzeugt wird.

Angeben von Speicheroptionen für Ressourcen

Sie können bestimmen, wie der Speicherplatz der Ressource verwaltet wird. Wenn Sie ein Neuling im Programmieren von Windows sind, werden Sie die Standardeinstellungen benutzen wollen.

Um Speicheroptionen anzugeben, markieren Sie die Ressource im Projektfenster und wählen *Ressource/Speicheroptionen*. Das Dialogfenster Speicheroptionen für Ressource wird angezeigt.

Schalten Sie jede Option aus, die Sie nicht anwenden wollen (Eine Beschreibung der vier Optionen erfolgt in Tabelle 14.1)

Wenn Sie bestimmte Speicheroptionen für eine Symbolressource setzen, gelten die Optionen für alle Bilder in der Ressource.

Folgende Speicheroptionen stehen für Ressource Dialogfenster zur Verfügung:

Tabelle 14.1: Speicheroptionen für die Ressource

Option	Beschreibung
Load On Call	Die Ressource wird nur bei Bedarf in den Arbeitsspeicher geladen. Mit Load On Call kann die Wartezeit bei Aufruf der Anwendung verkürzt werden. Deaktivieren Sie diese Option, wird Preload aktiviert, womit bei Aufruf der Anwendung die Ressource mit in den Arbeitsspeicher geladen wird. Preload ist sinnvoll, wenn Ihre Anwendung die Ressource von Anfang an benötigt.
Moveable	Windows verschiebt die Ressource im Arbeitsspeicher, um Platz für andere Speicherverwendung zu schaffen. Ist die Option ausgeschaltet, belegt die Ressource durchgängig einen festgelegten Speicherblock.
Discardable	Die Ressource wird nach Benutzung gegebenenfalls aus dem Arbeitsspeicher entfernt und bei neuer Benutzung wieder geladen. Ist die Option ausgeschaltet, kann Windows die Cursor-Ressource nicht aus dem Arbeitspeicher entfernen, solange Ihre Anwendung läuft. Ist zudem Pure ausgeschaltet, können Sie die Ressource aus der Anwendung heraus verändern. Das Ausschalten der Option ist zu RC.EXE nicht kompatibel.
Pure	Pure verhindert, daß die Ressource im Arbeitsspeicher verändert wird. Normalerweise ist diese Option immer eingeschaltet. Zu dieser Option können Sie mehr in der Windows Dokumentation erfahren.

Bezeichner verwenden

In Windows muß jede Ressource mit einem einmaligen Namen oder einer einmaligen Integernummer identifiziert werden können (Ressourcen ID genannt). Resource Workshop weist standardmäßig jeder neuen Ressource einen solchen Namen zu, z.B. DIALOG_1 für ein Dialogfenster oder MENU_1 für ein Menü.

Der Standardname sagt wenig über die Ressource aus, und der Bezug auf Ressourcen nur durch Namen vermindert die Effizienz Ihrer laufenden Anwendung.

Um diese Nachteile auszuschließen, können Sie die Ressource umbenennen und ihr einen *Bezeichner* zuweisen (mit C **#define** oder einer Pascal Konstante). Die Verwendung von Bezeichnern führt zu aussagekräftigeren Namen und steigert die Effizienz Ihrer laufenden Anwendung.

- Sie können der Ressource einen Integerwert zuweisen. Die Verwendung eines Integerwerts führt zur Verbesserung der Laufzeiteffizienz, ohne besonders aussagekräftig zu sein.(Mit DIALOG:225 wissen Sie nicht unbedingt, welches Dialogfenster damit gemeint ist).
- Sie können die Ressource umbenennen, ohne ihr einen Bezeichner zuzuordnen. Das Ergebnis ist eine Ressource mit einem aussagekräftigen Namen und einem Quelltext, mit dem einfacher gearbeitet werden kann als mit einem Bezeichner (Windows erwartet einen far Zeiger auf einen **char**), aber die Effizienz Ihrer laufenden Anwendung ist etwas geringer.

Bezeichnerkomponenten

Ein Bezeichner besteht aus zwei Teilen: einer Zeichenkette (dem Bezeichnernamen) und einem Wert (normalerweise eine IntegezZahl). Im folgenden Beispiel wird der Bezeichner mit dem Namen dlg_OpenFile und dem Wert 100 deklariert:

```
#define dlg_OpenFile 100
```

Im Projektfenster erscheint die Ressource als DIALOG:dlg_OpenFile und ist somit einfach als Open-File-Dialogfenster zu erkennen.

Innerhalb eines Ressourcen-Typs dürfen Bezeichner nur einmal vorkommen. Dabei werden nur die ersten 31 Zeichen zur Identifizierung verwendet; die weiteren Zeichen werden von Resource Workshop überlesen. Sie können die Bezeichnerwerte in Textfenstern ändern. Im Menüeditor können Sie z.B. für die ID eines Elements CM_FILENEW=125 eingeben. Bei Verlassen des Feldes wird der Wert des CM_FILENEW Bezeichners auf 125 gesetzt.

Ist ein Integerwert dem Bezeichner zugewiesen, erfolgt der Aufruf der Ressource während der Laufzeit schneller. Sie können jedoch nicht den Integerwert direkt als Parameter verwenden. Sie müssen zuvor entweder den Typ des Integerwerts in einen long Zeiger auf **char** umwandeln oder ein Makro verwenden.

Ist Ihr Programm in C bzw C++, können Sie das MAKEINTRESOURCE Makro verwenden. Ist Ihr Programm in Pascal verfaßt, können Sie den MakeIntResource Typ verwenden (Zeiger auf char).

Das Makro MAKEINTRESOURCE sieht aus wie ein Funktionsaufruf, führt aber eine Typumwandlung des Bezeichners durch. Um z.B.

dlg_OpenFile als Parameter in einem C/C++ Programm zu verwenden, geben Sie folgenden Ausdruck ein:

```
MAKEINTRESOURCE(dlg_OpenFile)
```

➡ Bei Bearbeitung einer .RES, .DLL oder ausführbaren Datei werden die IDs der Ressourcen in der Datei zu Integerwerten zurückübersetzt (decompiliert). Sie können bei diesem Dateityp keine Bezeichner zufügen, aber Sie können die Datei als .RC Datei speichern und dann den Ressourcen Bezeichner zuweisen. Mehr dazu im Abschnitt »Arbeiten mit Binärdateien«.

Bezeichner-dateien

Sobald Sie ein neues Projekt öffnen, sollten Sie zuerst die Datei angeben, in der Sie Bezeichner speichern wollen.

Speichern Sie Bezeichner in einer oder mehreren Header-Dateien (.h), in denen #define Anweisungen für die Zuweisung von Werten zu Bezeichnernamen verwendet werden.

In diesem Benutzerhandbuch werden Header-Dateien, Units und Include Dateien als Bezeichnerdateien bezeichnet.

Sie können einen Editor oder ein Textverarbeitungsprogramm benutzen, um Bezeichnerdateien zu erstellen. Auch mit Resource Workshop können Bezeichnerdateien erzeugt werden. Das wird im nächsten Abschnitt und im Abschnitt »Bezeichnerdateien hinzufügen« beschrieben.

Bezeichnerdateien erzeugen

Nach Öffnen des Projekts (*Datei/Neues Projekt*) und seiner Benennung (*Datei/Projekt speichern*) kann eine Bezeichnerdatei wie folgt zugefügt werden:

1. Wählen Sie *Datei/Dem Projekt hinzufügen*. Es erscheint das Datei einem Projekt hinzufügen Dialogfenster.
2. Klicken Sie auf die abwärts zeigende Pfeiltaste rechts neben dem Dateityp-Feld, damit eine Liste von Dateitypen aufgeklappt wird.
3. Schreiben Sie eine C-Anwendung, wählen Sie

   ```
   H Header-Datei
   ```

4. Ins Eingabefeld Dateiname geben Sie einen Namen für die Bezeichnerdatei an.
5. Mit Anklicken von *OK* wird das Datei einem Projekt hinzufügen Dialogfenster verlassen und gleichzeitig die Bezeichnerdatei erstellt.

C Header-Dateien

Ist Ihre Anwendung ein C-Programm, speichern Sie die Bezeichner in Header-Dateien mit der Erweiterung .H.

Die Anweisung **#define** weist den Bezeichnernamen Integerwerte zu. Sehen Sie hier ein Beispiel einer typischen Header-Datei:

```
/**************************************************
* Augewählte #define Anweisungen in RWCDEMOC.H  *
**************************************************/
#define bmp_StatusBar 101
#define cm_About_CUA 145
#define id_ClearWindow 229
#define dlg_About 104
#define dlg_FileNew 106
#define sth_Edit 15
#define men_Main 100
#define acc_Main 100
#define ico_RWCDemo 100
#define sth_EditClear 13
#define ScribbleWindow 100
#define Filewindow 101
#define GraphWindow 102
```

Neben **#define** Anweisungen kann eine Header-Datei auch Typ- und Strukturdefinitionen, Programmtext und Kommentare enthalten. Alle Daten in der Header-Datei, die nicht **#define** Anweisungen oder Präprozessor Kommandos sind, werden von Resource Workshop überlesen.

Automatische Bezeichnerverwaltung

Resource Workshop erzeugt und löscht automatisch Bezeichner für Sie. Um die automatische Bezeichnerverwaltung zu aktivieren, wählen Sie *Datei/Vorgaben* und markieren Sie Bezeichner automatisch erzeugen. Wenn Sie AppExpert von Borland C++ IDE aus benutzen, ist die Option eingeschaltet und kann nicht ausgeschaltet werden. Sie können diese Option hier nicht ausschalten, da die Funktionsfähigkeit von AppExpert von der Syntax automatischer Bezeichner abhängt.

Wenn Sie ein Ressourcenelement erzeugen, welches einen Bezeichner benutzt (z.B. Menüelemente), wird bei eingeschalteter automatischer Bezeichnerverwaltung ein einmaliger Bezeichner für das Element erzeugt und in die Header-Datei für diese Ressource (.RH oder .RC) eingetragen. Wenn Sie ein Element löschen, wird der Benutzer automatisch auch gelöscht.

Resource Workshop benutzt Präfixe für den Bezeichner, die verändert werden können. Im folgenden eine Liste der Standard Präfixe und der Menükommandos, mit denen sie verändert werden können. (Beachten Sie, daß Sie den Ressourcen-Typ bearbeiten müssen, um das korrekte Menü angezeigt zu bekommen.)

Tabelle 14.2: Bezeichner Präfixe

Ressource	Präfix	Menü-Kommandos
String Tabelle	IDS_	String Tabelle/Bezeichner Präfix ändern
Menü	CM_	Menü/Bezeichner Präfix ändern
Tastenkürzel	CM_	Tastenkürzel/Bezeichner Präfix ändern
Dialogfenster	IDC_	Optionen/Bezeichner Präfix ändern

Ohne Bezeichnerdatei arbeiten

Wenn Sie in Ihr Projekt keine Bezeichnerdatei aufnehmen, können Sie denoch Bezeichner für Ihre Ressourcen erstellen. Resource Workshop speichert die Bezeichner dann in der aktiven Projektdatei als **#define** Anweisungen.

Die Vorgehensweise ist aus folgenden Gründen *nicht* empfehlenswert:

- Wollen Sie später eine gesonderte Bezeichnerdatei erstellen, müssen Sie mit dem Texteditor die **#define** Anweisungen vom Ressourcen-Script in die Bezeichnerdatei übertragen.
- Da im Ressourcen-Script nur **#define** Anweisungen verwendet werden dürfen, müssen Sie zusätzliche Arbeit aufwenden, wenn Sie eine Pascal Unit oder Konstanten Datei erstellen wollen.

Bezeichner zufügen

Sie haben zwei Möglichkeiten, Bezeichner Ihrem Projekt zuzufügen.

- durch Umbenennen der Ressource.
- mit dem Bezeichner-Dialogfenster.

Umbenennen der Ressource.

Sie können Ressourcen beim Erzeugen umbennen. Findet Resource Workshop den neuen Ressourcennamen, werden Sie automatisch gefragt, ob Sie einen neuen Bezeichner für den Namen erzeugen wollen.

Haben Sie zum Beispiel ihrem Projekt eine neue Menü-Ressource hinzugefügt, gäbe Resource Workshop der Ressource den Namen MENU_1 (angenommen, es wäre das erste Menü im Projekt). Weil das Menü Dateien betrifft, entscheiden Sie sich, den Namen in MENU_FILE umzubenennen. Sie wählen dazu *Ressource/Umbenennen* und das Resource-umbenennen-Dialogfenster wird angezeigt.

Haben Sie den neuen Namen im Resource-umbenennen-Fenster eingetragen und auf *OK* geklickt, werden Sie gefragt, ob Sie einen neuen Bezeichner für diesen Namen erzeugen wollen. Wenn Sie auf *OK* klicken, zeigt Resource Workshop das Neue-Bezeichner-Dialogfenster.

Stellen Sie zuerst sicher, daß im Dateifeld der Name der Bezeichnerdatei eingetragen ist. Geben Sie dann einen Integerwert in das Wert Eingabefeld ein und klicken Sie auf *OK*. Sobald Sie Ihr Projekt sichern, wird der Bezeichner-Name und -Wert in Ihrer .H-, .PAS- oder .INC-Datei gespeichert.

Verwenden des Bezeichner-Dialogfensters

Mit Hilfe des Bezeichner-Dialogfensters können Sie einen Bezeichner Ihrem Projekt hinzufügen, bevor Sie die dazugehörige Ressource erzeugen. So fügen Sie einen Bezeichner hinzu:

1. Wählen Sie *Ressource/Bezeichner* und lassen Sie sich das Bezeichner Dialogfenster anzeigen.
2. Klicken Sie auf die *Neu* Taste. Es erscheint das Neuer-Bezeichner-Dialogfenster.
3. Benutzen Sie das Datei-Listenfeld, um die Datei zu bestimmen, in der der Bezeichner gespeichert werden soll.
4. Geben Sie den Ressourcennamen im Eingabefeld Name an.
5. Geben Sie den Integerwert im Eingabefeld Wert ein.
6. Klicken Sie auf *OK*.
 Der neue Ressourcen Name wird im Bezeichner Listenfeld angezeigt, und der Wert wird als (nicht verwendet) angegeben.

Mit dem Bezeichner-Dialogfenster können Sie nicht nur Bezeichner hinzufügen, sondern auch Bezeichner bearbeiten, löschen oder sich auflisten lassen und den Ressourcen-Editor aufrufen. Mit *Ressource/Bezeichner* können Sie sich das Bezeichner Dialogfenster anzeigen lassen.

Bezeichner bearbeiten

Sie können den Bezeichnerwert in folgenden Schritten ändern:

1. Mit *Ressource/Bezeichner* wird das Bezeichner Dialogfenster angezeigt.
2. Markieren Sie den Bezeichner, dessen Wert Sie ändern wollen.
3. Mit Klicken auf die Ändern Taste wird das Wert des Bezeichners ändern Dialogfenster angezeigt.
4. Geben Sie den neuen Wert in das Neuer Wert Eingabefeld ein und klicken Sie auf *OK*.

Der neue Bezeichner Wert wird in Ihrer .H, .PAS oder .INC Datei gespeichert sobald Sie *Datei/Projekt speichern* wählen.

Bezeichner löschen

Wenn ein Bezeichner im Projekt nicht benutzt wird, sollten Sie ihn aus der .H, .Pas oder .INC Datei löschen. Einige Möglichkeiten, wie unbenutzte Bezeichner entstehen können:

- Sie haben einem Bezeichner einer Ressource zugewiesen und dann die Ressource gelöscht.
- Sie haben einen Bezeichner dem Projekt zugefügt und ihn dann nie benutzt.

- Sie haben eine Ressource umbenannt, die bereits einen Integerwert zugewiesen hatte.

So löschen einen Bezeichner:

1. Zeigen Sie mit *Ressource/Bezeichner* das Bezeichner Dialogfenster.
2. Markieren Sie den zu löschenden Bezeichner.
 Wenn der markierte Benutzer nicht zu einer bestimmten Ressource gehört (da entweder die Ressource gelöscht wurde oder der Bezeichner nicht gebraucht wurde), zeigt das Verwendung Fenster (nicht verwendet) an.

Auch ein noch benutzter Bezeichner *kann* gelöscht werden.

Ist allerdings der Bezeichner noch mit einer Ressource verbunden, wird im Verwendung Fenster automatisch der Typ und der Name der Ressource hervorgehoben.

3. Klicken Sie auf die *Löschen* Taste.

Ist der Bezeichner unbenutzt, wird er sofort gelöscht. Kein Fenster mit Warnmeldungen wird angezeigt.

Wird der Bezeichner noch benutzt, wird ein Fenster mit einer Warnmeldung ausgegeben. Durch Klicken auf *Ja* löschen Sie den Bezeichner. Mit Klicken auf *Nein* wird der Bezeichner nicht gelöscht.

4. Wenn Sie das nächste Mal *Datei/Projekt speichern* wählen, wird die Bezeichnerdatei auf den neuesten Stand gebracht und gelöschte Bezeichner somit entfernt.

Bezeichner auflisten

So listen Sie die Bezeichner in Ihrem Projekt auf:

1. Zeigen Sie mit *Ressource/Bezeichner* das Bezeichner Dialogfenster an.
2. Wählen Sie entweder Alles oder Einzelne Datei in der Gruppe Darstellen.
3. Wenn Sie Einzelne Datei wählen, markieren Sie die Datei, deren Bezeichner Sie sich ansehen wollen, im Listenfenster für Dateien in der Gruppe Darstellen.

Starten eines Ressourceneditors

Sie können das Bezeichner-Dialogfenster nutzen, um einen Ressourcen-Editor aufzurufen, der schon eine Ressource zum Bearbeiten geladen hat.

So starten Sie den Ressourcen-Editor im Bezeichner-Dialogfenster:

1. Suchen Sie im Bezeichner Feld, bis die gewünschte Ressource markiert ist. Typ und Name der Ressource werden im Verwendung Feld angezeigt.
2. Doppelklicken Sie auf den hervorgehobenen Typ und Namen im Feld Verwendung. Der Editor wird aufgerufen und hat die Ressource bereits geladen.

Vorgaben einstellen

Mit *Datei/Vorgaben* können Vorgaben eingestellt werden. Die Online Hilfe bietet hier zusätzliche Informationen zu Punkten, die in diesem Abschnitt nicht behandelt werden. (Klicken Sie dazu auf die Hilfe Taste im Vorgaben Dialogfenster.

Stufenweises Rückgängigmachen

Resource Workshop erlaubt Ihnen in beliebigen Ressourcen-Editoren,mit Rückgängig und Widerrufen Aktionen in mehreren Stufen rückgängig zu machen. Abhängig vom verfügbaren Speicherplatz in Ihrem Computer können bis zu 99 Aktionen rückgängig gemacht oder erneut ausgeführt werden. Die Standardeinstellung ist dabei 10 Ebenen.

Mit [*Alt*]+[*Rück*] oder *Bearbeiten/Widerrufen* veranlassen Sie das Rückgängigmachen. Mit [*Shift*]+[*Alt*]+[*Rück*] wird erneut ausgeführt. Sie können Ihre gesamte Sitzung auf diese Weise vorwärts und rückwärts in so vielen Stufen durchlaufen, wie es voreingestellt ist.

Die Anzahl der Ebenen kann durch die verfügbare Speicherkapazität eingeschränkt sein. Eine Ressource, wie z.B. ein großes Bitmap mit mehreren Füllmustern, kann bewirken, daß weniger Ebenen bearbeitbar sind.

Texteditor

Beim Laden eines Projekts werden alle Ressourcen der Datei compiliert. Wird dabei ein Fehler entdeckt, wird die Compilierung abgebrochen. Sie werden auf den Fehler hingewiesen und gefragt, ob Sie die Datei mit dem in der Option bestimmten externen Texteditor bearbeiten wollen

Der externe Texteditor ist standardmäßig der Windows Notizblock. Ein anderer von Ihnen gewählter Editor muß unter Windows lauffähig sein. (Ein DOS-Editor mit .PIF Datei genügt.)

Mehrfachspeichern

Die Vorgaben .RES und Ausführbar steuern, wie ein Projekt gespeichert wird, wenn Sie *Datei/Projekt speichern* wählen. Die Vorgaben sind nur verfügbar, wenn ein Ressourcencompiler geöffnet worden ist, da die Vorgaben nur für ein bestimmtes Projekt gelten. Unabhängig von den Einstellungen zu Mehrfachsichern wird das Projekt auch immer im ursprünglichen Format gespeichert. (Ist zum Beispiel das Projekt eine .RC Datei, werden die Ressourcen der Datei als Ressourcen-Scripts gespeichert, zusätzlich zu jeder möglichen Vorgabe zur Mehrfachsicherung.)

- *.RES*. Compiliert die Ressourcen des aktuellen Projekts und sichert sie im binären .RES Format.
- *Ausführbar*. Compiliert die Ressourcen des aktuellen Projekts und bindet sie in die ausführbare Datei ein, die in dieser Option angegeben ist (Das kann eine .EXE oder .DLL Datei sein.).

- *Backup Dateien erzeugen* Wenn Sie diese Option einschalten, wird beim Speichern ein Satz von Sicherungsdateien zusätzlich erzeugt. Sicherungsdateien haben eine Tilde (~) als erstes Zeichen der Dateinamenerweiterung. Speichern Sie zum Beispiel MYPROJ.BMP, heißt die Sicherungsdatei MYPROJ.~BM.

Windows Version

Mit diesen Tasten wird bestimmt, unter welcher Windows Version Ihre Ressourcen eingesetzt werden sollen. Beachten Sie, daß .RES und .EXE Dateien für Windows 3.1 nicht unter Windows 3.0 verwendet werden können.

Wenn die .RES Datei für Windows 3.1 bestimmt ist, können Sie nicht die Windows 3.0 Version von RC.EXE zum Einbinden der Ressourcen in Ihre Anwendung verwenden. Sie müssen dazu das Borland Dienstprogramm oder die Version 3.1 des Microsoft Ressourcencompilers einsetzen.

Diese Option steht nur zur Verfügung, wenn gerade kein Projekt geöffnet ist.

Arbeiten mit Binärdateien

Mit Resource Workshop können ausführbare Dateien, .RES und .DLL Dateien als Projekte geöffnet werden, so daß Sie Ihre Benutzerschnittstelle neu gestalten können. Möglicherweise wollen Sie z.B. die Benutzerschnittstelle Ihrer Anwendung in eine Fremdsprache übertragen.

Wenn Sie eine dieser Dateien öffnen, rückübersetzt (decompiliert) Resource Workshop die Ressourcen. Sie werden so dargestellt, als ob sie Teil einer gewöhnlichen .RC Datei wären. Nachdem Sie Ihre Änderungen gemacht haben, werden die Dateien wieder in Binärform übersetzt und in der ursprünglichen Datei gespeichert.

Alle Ressourcen IDs in Binärdateien müssen Integerzahlen sein.

Da die Ressourcen in einer decompilierten Binärdatei nicht in Ressourcen Script-Form gespeichert sind, können Sie den Ressourcen IDs keine Bezeichner zuweisen.

Sie können das Projekt jedoch als .RC Datei speichern. Die Ressourcen lassen sich dann als Ressourcen-Scripts speichern, und Sie können Ihnen Bezeichner zuweisen.

Wenn Sie die Benutzerschnittstelle eines Programms anpassen möchten und nur über die ausführbare Datei oder DLL verfügen, möchten Sie dennoch eventuell Ihre Änderungen in einer separaten .RC Datei speichern, damit Sie sie für die nächste Version Ihres Programms verwenden können. Die Ressourcen in Ihrer .RC Datei müssen dieselben Ressourcen IDs wie ihre Gegenstücke in der neuen Version haben. Außerdem müssen sie kompatibel mit der neuen Version sein.

Wenn Sie das Projekt als .RC Datei speichern, speichert Resource Workshop die Ressourcen nicht automatisch wieder in der Originaldatei. Dies erfolgt nur, wenn Sie den Originalnamen der Datei im Vorgaben Dialogfenster als Mehrfachspeicher-Option eingegeben haben.

Um eine Binärdatei als Projekt zu speichern und Bezeichner hinzuzufügen, führen Sie fol zu speichern und Bezeichner hinzuzufügen, führen Sie folgende Schritte aus:

1. Wählen Sie *Datei/Projekt Öffnen* und wählen im Dialogfenster Projekt Öffnen die ausführbare, .RES- oder DLL-Datei.
2. Wählen Sie *Datei/Projekt Speichern unter*. Wählen Sie im Dialogfenster Datei Speichern unter aus dem Listenfeld Dateityp RC Ressourcen-Script. Geben Sie den Namen der neuen .RC Datei ein.

 Wenn Sie *OK* klicken, um die Datei zu speichern, bringt Sie Resource Workshop automatisch in die .RC Datei.
3. Wählen Sie *Datei/Vorgaben* und geben den Namen der Original-Binärdatei als Mehrfachspeicher-Option ein.

 Wenn die Original-Binärdatei eine .RES Datei war, markieren Sie .RES und geben den Namen in dieses Textfeld ein.

 Wenn die Original-Binärdatei eine ausführbare Datei oder eine DLL war, markieren Sie Ausführbare und geben den Namen in dieses Textfeld ein.
4. Wählen Sie *Datei/Dem Projekt hinzufügen* und geben eine Bezeichnerdatei ein, die die neuen Bezeichner aufnehmen soll. Wenn die angegebene Datei nicht existiert, erzeugt Resource Workshop eine neue.

Machen Sie unbedingt eine Sicherungskopie der aktuellen Integerwerte der Ressourcen IDs.

5. Nehmen Sie Ihre Änderungen an den Ressourcen vor und geben an den gewünschten Stellen Bezeichner an. Für jeden neuen Bezeichner fragt Sie Resource Workshop, ob er in der Bezeichnerdatei gespeichert werden soll.
6. Wenn Sie aufhören und die Datei speichern, speichert Resource Workshop sowohl die .RC Datei als auch die Binärdatei. Wenn die Binärdatei eine ausführbare Datei oder DLL ist, sind die geänderten Ressourcen damit verbunden und stehen sofort zur Verfügung, wenn Sie das Programm ausführen.

32-Bit-Ressourcen erzeugen

Standardmäßig erzeugt Resource Workshop 16-Bit-Ressourcen, wenn Sie unter 16-Bit-Windows arbeiten, und 32-Bit-Ressourcen, wen Sie unter NT

arbeiten. Sie können den Typ der zu erzeugenden Ressourcen ändern. Wenn Sie beispielsweise unter NT arbeiten, können Sie auch 16-Bit-Ressourcen erzeugen, und wenn Sie unter Windows arbeiten, können Sie auch 32-Bit-Ressourcen erzeugen.

Um den Typ des zu erzeugenden Ressourcenprojekts zu ändern, schließen Sie alle offenen Projekte. Gehen Sie anschließend wie folgt vor:

1. Wählen Sie *Datei/Vorgaben*.
2. Markieren Sie die gewünschte Windows-Version und klicken *OK*.
3. Wählen Sie *Datei/Neues Projekt*. Sobald Sie ein Projekt erzeugt haben, können Sie den Zieltyp nicht mehr ändern (so können Sie beispielsweise nicht von Win32 zu Windows 3.1 wechseln).

Sie können außerdem im Dialogfenster *Datei/Vorgaben* eine Haupt- und Zweitsprache wählen, wenn Sie als Ziel Win32 angegeben haben. Beachten Sie, daß die hier gewählte Sprache die in NT verwendete sein muß, da Ihre Ressourcen sonst eventuell nicht korrekt angezeigt werden. Wenn Sie beispielsweise Französisch und Deutsch als Ressourcensprache wählen und NT auf Englisch steht, können Sie zwar Ihre Ressourcen in Resource Workshop erzeugen, doch werden sie beim Ablauf Ihres Programms eventuell nicht korrekt angezeigt. Dies gilt insbesondere für Menü-Ressourcen.

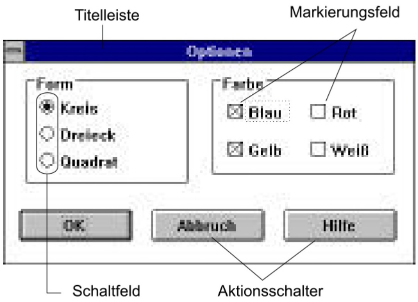
K A P I T E L 15

Dialogfenster erzeugen

Mit Hilfe von *Dialogfenstern* kann der Benutzer mit der Anwendung kommunizieren. Ein Dialogfenster ist ein Popup-Fenster, in das der Benutzer Informationen eingeben kann (zu öffnende Dateien, anzuzeigende Farben, zu suchenden Text usw.).

Das Dialogfenster enthält meistens einige Dialogelemente, wie Tasten, Textfelder und Bildlaufleisten. Dialogelelemente erlauben dem Benutzer, Informationen einzugeben, können aber auch benutzt werden, um unveränderbare Texte oder Grafiken in einem Dialogfenster anzuzeigen.

Abbildung 15.1: Ein typisches Dialogfenster



Der Aufruf des Dialogeditors

Mit Hilfe des Dialogeditors können Sie neue Dialogfenster erzeugen oder bestehende Dialogfenster bearbeiten. Wie Sie den Dialogeditor aufrufen, hängt davon ab, welche Aufgabe Sie ausführen wollen.

Eine neues Dialogfenster erzeugen

Um ein neues Dialogfenster zu erzeugen,

1. Wählen Sie *Datei/Neues Projekt* um ein neues Projekt zu öffnen oder *Datei/Projekt öffnen* um ein bestehendes Projekt zu laden.

In Kapitel 14 wird beschrieben, wie Sie ein Projekt öffnen.

2. Mit *Ressource/Neu* erscheint das Neue Ressource Dialogfenster.
3. Im Feld Ressourcen-Typ doppelklicken Sie auf DIALOG.
4. Ein Dialogfenster erscheint, in dem Sie aufgefordert werden, eine Vorlage für Ihr Dialogfenster anzugeben. Wählen Sie einen Dialog-Typ und klicken Sie auf OK.
5. Der Dialogeditor erscheint mit der Dialog Vorlage des Typs, den Sie in Schritt 4 angegeben haben. In Abbildung 15.2 sehen Sie die Bestandteile des Dialogeditors abgebildet.

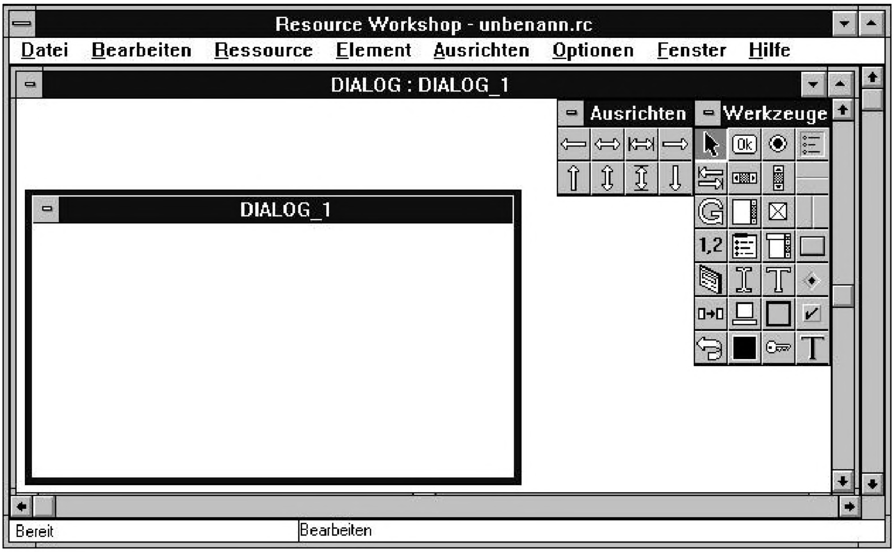


Abbildung 15.2: Dialogeditor mit leerem Dialogfenster

Bearbeiten eines bereits existierenden Dialogfensters

Um ein bereits in einer Projektdatei bestehendes Dialogfenster zu bearbeiten,

1. öffnen Sie das Projekt, welches das Dialogfenster enthält, das Sie bearbeiten wollen.
2. Suchen Sie das Dialogfenster, das Sie bearbeiten wollen, im Projektfenster und doppelklicken Sie auf dessen Namen.
3. Die Projektdatei erscheint wie in Abbildung 15.2, nur daß statt der Vorlage nun Ihr gewähltes Dialogfenster zu sehen ist.

Benutzung des Dialogeditors

Wenn Sie ein neues Dialogfenster erzeugen wollen, wird Ihnen der Dialogeditor eine leere Vorgabe anzeigen. Sie können nun den Typ Ihres Fenster festlegen, einen Titel bestimmen, wählen, welche Schrift darin angezeigt wird, usw. Sie können ebenfalls Dialogelemente hinzufügen, verändern, anordnen, umgruppieren, bewegen, vergrößern, verkleinern und löschen, so daß Ihr Dialogfenster genauso aussieht, wie Sie es haben möchten.

Ein Dialogfenster auswählen

Um Veränderungen im Dialogfenster zu machen oder dessen Eigenschaften zu bestimmen, müssen Sie es zuvor *markieren*. Um ein Dialogfenster zu markieren, klicken Sie in die Titelleiste.

Ist das Dialogfenster markiert, wird ein *Markierungsrahmen* um das Fenster angezeigt, und Sie können

- das Dialogfenster mit Hilfe der Maus oder der Pfeiltasten bewegen.
- die Größe des Dialogfensters auf eine der folgenden Arten verändern:
 - Gehen Sie mit dem Mauszeiger an den Rahmen oder in eine Ecke. Halten Sie die Maustaste gedrückt und ziehen Sie das Fenster zur gewünschten Größe.
 - Markieren Sie das Dialogfenster durch Klicken in der Titelleiste. Mit *Ausrichten/Größe* wird das Dialogfenster Größe Dialog angezeigt. Sie können die horizontale (CX) und vertikale (CY) Größe des Dialogfensters in Dialogeinheiten eingeben. (Eine y-Dialogeinheit ist 1/8 der Schrifthöhe und eine x-Dialogeinheit ist1/4 der Schriftbreite.) Die eingegebenen Größenangaben beziehen sich auf den Gesamtumfang des Dialogfensters.

Einstellen der Dialogfenster-Attribute

Mit Hilfe des Dialogfensters Fenstereigenschaften können Sie die Eigenschaften Ihres neuen Dialogfensters festlegen, einschließlich

- Titel, Klasse, und Menü
- Fenstertyp
- Rahmenart
- Eigenschaften
- Schriftart

Um sich das Dialogfenster Fenstereigenschaften anzeigen zu lassen, wählen Sie das Dialogfenster, für das Sie Eigenschaften bestimmen wollen, und drücken Sie die [*Eingabetaste*]. Oder Sie doppelklicken auf die Titelleiste des Dialogfensters, das Sie bearbeiten wollen (der Mauszeiger erscheint als Pfeil).

Titel hinzufügen

Sie können auch Titel ändern, indem Sie das Eigenschaften Feld verwenden (Wählen Sie *Optionen/Eigenschaften* anzeigen).

Um einen Titel hinzuzufügen,

1. Doppelklicken Sie in die Titelleiste des Dialogfenster und das Fenster Fenstereigenschaften erscheint.
2. Geben Sie den neuen Namen des Dialogfensters in das Titelfeld ein.
3. Unter Rahmenart muß Titel eingestellt sein.
4. Klicken Sie auf *OK*.

Im Fenster Fenstereigenschaften können Sie den Fenstertyp bestimmen:

- Popup. Da die meisten Dialogfenster Popup-Fenstersind, ist dies die Voreinstellung.
- Untergeordnet. Ein Unterfenster eines anderen Fensters.
- Überlappend. Ein Popup-Fenster, das ganz oder teilweise von anderen Fenstern verdeckt werden kann. Diese Eigenschaft wird meistens dem Hauptfenster der Anwendung zugeordnet.

Ändern der Rahmenart

Die Rahmenart des Dialogfensters bestimmt die Erscheinungsform des Rahmens und der Titelleiste. Rahmenarten können wie folgt definiert werden:

- Dialograhmen. Zweifache Rahmenlinie ohne Titelleiste.
- Rahmen. Einfache Rahmenlinie ohne Titelleiste.
- Titel. Einfache Rahmenlinie und eine Titelleiste, in der ein Titel dargestellt werden kann (Standardvoreinstellung).
- Kein Rahmen. Keine Rahmenlinie und keine Titelleiste.

Eigenschaften ändern

Eigenschaften legen das Aussehen des Fensters fest und wie der Benutzer damit arbeiten wird. Sie können eine oder mehrere der folgenden Eigenschaften für Ihr Dialogfenster auswählen:

- Systemmenü. Es wird ein Systemmenüfeld auf der linken Seite der Titelleiste eingefügt. (Das erscheint nur, falls Sie auch Titel gewählt haben) Wenn das Dialogfenster ein untergeordnetes Fenster ist, erscheint das Feld zum Schließen des Fensters.
- Dicker Rahmen. Das Fenster wird mit einem dicken Rahmen umschloßen. Diese Option definiert,was der Benutzer sieht, wenn das Dialogfenster in einer Anwendung erscheint. Wenn das Dialogfenster vergrößer- oder verkleinerbar sein soll, müssen Sie diese Option benutzen. Diese Option sollte nicht mit der Option Dicker Rahmen im Kommando Voreinstellungen im Dialogeditor verwechselt werden. Die Option dort bestimmt, wie das Dialogfenster aussieht, wenn Sie es im Dialogeditor auswählen.

- Vert. Bildlaufleiste. Eine senkrechte Bildlaufleiste wird an den Rahmen des Dialogfenster angefügt.
- Horiz. Bildlaufleiste. Eine waagrechte Bildlaufleiste wird an den Rahmen des Dialogfenster angefügt.
- Symbolfeld. Fügt eine Taste auf die rechte Seite der Titelleiste ein, mit der das Fenster auf Symbolgröße verkleinert werden kann. Die Taste erscheint nur, wenn auch die Eigenschaft Titel gewählt wurde.
- Zoomfeld. Fügt eine Taste auf die rechte Seite der Titelleiste ein, mit der das Fenster auf Vollbild vergrößert werden kann. Die Taste erscheint nur, wenn auch die Eigenschaft Titel gewählt wurde.
- Absolute Koordinaten. Das Dialogfenster wird relativ zum Bildschirm und nicht am übergeordneten Fenster ausgerichtet.
- Modal. Das Dialogfenster wird zum modalen Fenster. Das heißt, der Benutzer kann solange nicht weiterarbeiten, bis das modale Fenster geschlossen wird.
- Lokaler Heap. Speicherplatzanforderungen durch die Eingabefelder des Fensters werden auf den lokalen Heap der Anwendung geleitet.
- Verwenden Sie diese Eigenschaft, wenn Sie die EM_SETHANDLE und EM_GETHANDLE Nachrichten benutzen.
- Verschiebbar. Eine Kombination von Titel- und Dialogfenstereigenschaften wird ausgewählt. Der Benutzer ist dann in der Lage, das Dialogfenster frei zu verschieben.
- Keine Idle-Botschaften. WM_ENTERIDLE Nachrichten werden nicht an das Hauptfenster der Anwendung weitergeleitet. Das Dialogfenster muß die Modal Eigenschaft besitzen, damit diese Option wirksam wird.
- Schutz (unterg.). Das Dialogfenster darf untergeordnete Fenster nicht verdecken.
- Schutz (gleichg.). Das Dialogfenster darf gleichgeordnete Fenster nicht verdecken. Das Zeichnen wird auf dieses Fenster beschränkt. Für Popup-Fenster muß diese Option nicht verwendet werden, kann aber bei untergeordneten Dialogfenstern sinnvoll sein.
- Sichtbar. Vor der Rückkehr aus CreateDialog wird ein nicht-modales Dialogfenster sichtbar. Diese Option hat keine Wirkung auf modale Dialogfenster, welche am häufigsten verwendet werden. In der Voreinstellung ist diese Option nicht ausgewählt (NOT WS_VISIBLE).

Schriften ändern

Standardmäßig wird als Schrift in Dialogfenstern MS Sans Serif in 8 Punkt Größe verwendet. Sie können die Schrift und die Größe jedoch ändern. Um eine Schrift für Ihr Dialogfenster auszuwählen,

1. öffnen Sie das Dialogfenster Eigenschaften.
2. Klicken Sie auf die Schriftentaste. Das Fenster Schrift wählen erscheint.

3. Im Schrift wählen Dialogfenster können Sie Größe und Schriftnamen für Ihren Text in Ihrem Dialogfenster auswählen. Die Buchstaben im Eingabefeld unten im Dialogfenster zeigen das Aussehen der aktuell gewählten Schrift. (Beachten Sie: Windows nimmt nur Text in Fettdruck für Dialogfenster an.)

Ein Menü einfügen

Weil ein Dialogfenster auch ein Fenster ist, kann es auch ein Menü enthalten. Manche Anwendungen benutzen ein Dialogfenster als Hauptfenster. In diesem Fall ist das Einfügen von Menüs sicher sinnvoll.

Um ein Menü Ihrem Dialogfenster hinzuzufügen,

1. erzeugen Sie ein Menü als gesonderte Ressource und fügen sie Ihrem Projekt hinzu (siehe Kapitel 16). Merken Sie sich den Ressourcenamen oder den ID-Wert, mit dem Sie dieses Menü ansprechen können.
2. Öffnen Sie den Dialogeditor und laden Sie das Dialogfenster, dem das Menü hinzugefügt werden soll.
3. Öffnen Sie das Fenster Eigenschaften.
4. In das Eingabefeld Menü tragen Sie den Namen oder die ID des Menüs ein.

➡ Der Dialogeditor zeigt das Menü im Normalfall nicht an. Wenn Sie das Dialogfenster testen, zeigt er ein generisches Menü »Menu« an, das nur einen einzigen Eintrag enthält.

Zuweisen einer selbstdefinierten Klasse an ein Dialogfenster.

Als erfahrener Windows Programmierer könnten Sie es vorziehen, Nachrichten des Dialogfensters mit eigenen Prozeduren statt den Standard Windows Prozeduren zu verarbeiten. Sie könnten auch wünschen, das Fenster zu einem Dialogfenster im Borland Stil zu machen. Dazu können Sie dem Dialogfenster eine selbstdefinierte Klasse zuweisen.
Um einer selbstdefinierten Klasse ein Dialogfenster zuzuweisen,

1. öffnen Sie das Dialogfenster Eigenschaften
2. Schreiben Sie den Namen der Klasse ins Eingabefeld Klasse

Festlegung der Dialogfensterposition

Wird In Ihrem Dialogfenster WS_OVERLAPPED verwendet (d.h. Sie haben Überlappend als Fenstertyp gewählt), können Sie es Windows überlassen, Ihr Dialogfenster auf dem Bildschirm zu positionieren. Diese Möglichkeit wird für gewöhnlich bei Anwendungen verwendet, für die das Dialogfenster das Hauptfenster ist. Um die Verwaltung der Fensterposition Windows zu überlassen,

1. markieren Sie durch Klicken auf die Titelleiste des Dialogfensterrahmens.

2. Wählen Sie *Ausrichten/Größe*.
3. Im Dialogfenster Dialoggröße klicken Sie auf Durch Windows gesetzt. Der Wert der X-Koordinate wird nun von Resource Workshop hellgrau dargestellt, um anzuzeigen, daß Windows die Verwaltung des Dialogfensters übernommen hat.
4. Mit Klicken auf OK verlassen Sie das Fenster Dialoggröße.

Arbeiten mit Dialogelementen

Dialogelemente sind die einzelnen Bestandteile des Dialogfensters. Benutzer können damit Informationen an Ihre Anwendung eingeben oder Informationen erhalten. Dialogelemente können in folgende allgemeine Kategorien unterteilt werden:

- Tasten
- Bildlaufleisten
- Listenfelder
- Eingabefelder
- Statische Dialogelemente
- Kombinationsfelder
- Benutzerdefinierte Dialogelemente

Dialogelementgruppen

Jedes der Dialogelemente kommt aus einer der folgenden Gruppen:

- Standard Windows Dialogelemente, wie Tasten, Markierungs- Listen- und Schaltfelder . Die Symbole für diese Dialogelemente finden Sie in der zweiten und dritten Spalte der Werkzeugpalette. Standard Windows Dialogelemente sind immer verfügbar.
- Dialogelemente im Borland Stil (BWCC) sind auch immer verfügbar. Diese Dialogelemente sind gegenüber den Standard Windows Dialogelementen grafisch und funktional erweitert worden und umfassen Schalter, Markierungs- und Schaltfelder. Symbole für Borland Dialogelemente finden Sie in der vierten Spalte der Werkzeugpalette. Eine genauere Beschreibung der Borland Dialogelemente finden Sie in Anhang B.

Benutzerdefinierte Dialogelemente, deren Klasse von Resource Workshop erkannt wird. Diese Dialogelemente sind in einer DLL (Dynamic Link Library) gespeichert, die die Funktion ListClasses enthält. Ist die DLL Datei installiert worden, erscheinen die Symbole der Dialogelemente in den Spalten rechts von den Borland Dialogelementen in der Wekzeugpalette. Diese Art von Dialogelementen ist auf Seite 29 beschrieben.

Benutzerdefinierte Dialogelemente, deren Klasse vom SDK Editor erkannt wird. Diese Dialogelemente sind ebenfalls in einer DLL gespeichert, die jedoch nicht die Funktion *ListClasses* enthält, sondern nur die

entsprechenden Bitmaps. Diese werden nicht in der Werkzeugpalette dargestellt, sondern nur Ihre Namen erscheinen in der Drop-down Liste des Dialogfensters Neues benutzerdefiniertes Element. Wenn Sie eines dieser Dialogelemente in Ihrem Dialogfenster verwenden, sehen Sie das dazugehörige Bitmap auf dem Bildschirm, wenn Resource Workshop im WYSIWYG-Modus betrieben wird.

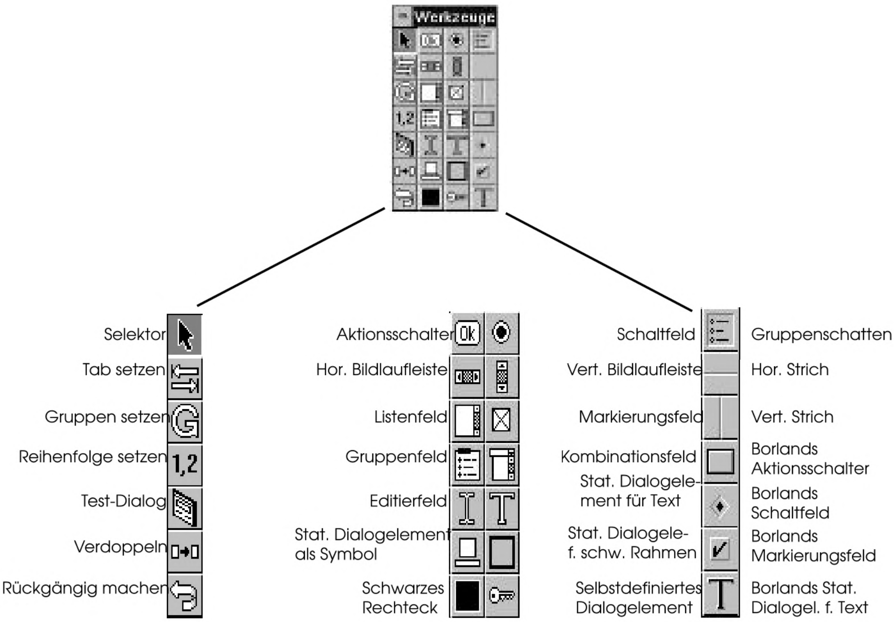
Benutzerdefinierte Dialogelemente, deren Klasse weder durch den SDK Editor noch durch Resource Workshop erkannt wird. Die Namen dieser Dialogelemente werden in die Drop-down Liste des Dialogfensters Neues benutzerdefiniertes Element angefügt. Sie erscheinen im WYSIWYG-Modus nur als graue Rechtecke auf dem Bildschirm.

Die Werkzeugpalette

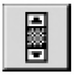
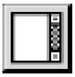
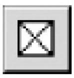
Die linke Spalte der Werkzeugpalette enthält Werkzeuge, mit denen der Arbeitsmodus des Dialogeditors eingestellt werden kann.

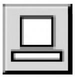
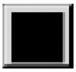
Die übrigen Spalten enthalten Symbole für die Standard Windows Dialogelemente, die Borland Dialogelemente, und sonstige Dialogelemente, die Sie geladen haben und die von Resource Workshop erkannt werden konnten.

Abbildung 15.3:
Die Werkzeugpalette



Hier eine Kurzbeschreibung der Standard Windows Dialogelemente, deren Symbole in der zweiten und dritten Spalte der Werkzeugtabelle zu sehen sind.

	Taste	Eine rechteckige Taste, die vom Benutzer »gedrückt« werden kann, um eine Aktion auszulösen. Da Tasten immmer Text enthalten, müssen Sie für jede einen Titel bestimmen.
	Schaltfeld	Eine runde Taste mit Text zur Linken oder Rechten. Ist die Taste markiert, erscheint ein dicker Punkt in der Mitte. Schaltfelder werden in Gruppen angeordnet, die ähnliche, sich aber gegenseitig ausschließende Optionen darstellten.
	Horizontale Bildlaufleiste	Ein waagrechtes Rechteck mit Pfeiltasten an jedem Ende.
	Vertikale Bildlaufleiste	Ein senkrechtes Rechteck mit Pfeiltasten an jedem Ende.
	Listenfeld	Ein Rechteck, das normalerweise mehrere Textketten auflistet. Wenn Sie einer Liste die Eigenschaft Ausgabe durch Anwendung geben, können auch grafische Darstellungen von Datenlisten angezeigt werden. Normalerweise kann ein Benutzer den Inhalt der Liste durchsehen und einen oder mehrere Einträge markieren. Sie sehen oft Listenfelder in Datei öffnen Dialogfenstern.
	Markierungsfeld	Eine rechteckige Taste mit Text zur Linken oder Rechten. Ist die Taste markiert, erscheint ein Kreuz im Rechteck. Ist die Taste nicht markiert, ist das Rechteck leer. Markierungsfelder werden oft verwendet, um logische (Eingeschaltet/Ausgeschaltet) Zustände für bestimmte Optionen darzustellen.
	Gruppenfeld	Ein rechteckiges Feld, mit dem andere Dialogelemente visuell zusammengefaßt werden können. Sie können einen Titel angeben, der in der linken oberen Ecke des Gruppenfeldes angezeigt wird.
	Kombinationsfeld	Eine Kombination aus Listen- und Eingabefeld oder Listenfeld und statischem Element.
	Eingabefeld	In dieses Rechteck kann der Benutzer mit Hilfe der Tastatur Text eingeben.
	Statischer Text	Text, der im Dialogfenster erscheint.

	Statisches Symbol	Ein Symbol.
	Schwarzer Rahmen	Ein leerer, rechteckiger Rahmen, der die Farbe des Rahmens des aktuellen Fensters hat.
	Schwarzes Rechteck	Ein statisches Symbol, das als Rechteck in der Farbe des Rahmens des aktuellen Fensters erscheint.
	Benutzerdefiniertes Dialogelement	Ein Dialogelement, das weder die Standard Windows noch Borland Klasse aufweist und von Resource Workshop nicht erkannt wird.

Dialogelemente markieren

Für das Bearbeiten eines Fensters müssen manchmal Dialogelemente *markiert* werden. Zum Beispiel müssen Sie ein Element markieren, wenn Sie seine Größe verändern wollen, und Sie müssen zwei Elemente markieren, wenn Sie sie relativ zueinander ausrichten wollen. Ein markiertes Element ist von einem Markierungsrahmen umgeben.



Um zum *Auswahlmodus* zu wechseln, in dem Sie Dialogelemente markieren können, klicken Sie auf das Auswahlwerkzeug (das Pfeilsymbol). Der Mauszeiger erscheint nun selbst als Pfeil.

Um ein einzelnes Element zu wählen, klicken Sie mit diesem Pfeil an den Rand des Elements oder auf das Element selbst. Dies hängt vom Schaltfeld Auswahl in Rahmennähe ab, das sich im Dialogfenster Optionen/Vorgaben unter Auswahloptionen befindet.

Wenn Sie mehr als ein Dialogelement markieren wollen, haben Sie zwei Möglichkeiten:

- Sie ziehen einen *Markierungsrahmen* wie folgt über diejenigen Dialogelemente, die sie markieren wollen:

1. Markieren Sie das Auswahlwerkzeug.
2. Klicken Sie auf ein Element. So wird sichergestellt, daß ein Element und nicht das gesamte Dialogfenster markiert wird.
3. Ziehen Sie ein Markierungsrechteck. Abhängig vom Zustand des Schaltfeldes Auswahl durch umschließendes Rechteck im Dialogfenster Optionen/Vorgaben, muß der Markierungsrahmen entweder ganz die Elemente umschließen oder sie nur berühren.
 Die gewählten Dialogelemente sind von einem Markierungsrahmen umgeben. Wenn Sie den Rahmen ziehen, ändern Sie die Position der markierten Dialogelemente.

- Sie können [*Umschalt*]-Klicken, um Dialogelemente zu markieren, die durch das Ziehen eines Markierungsrahmens nicht zu erreichen wären. (So kann es zum Beispiel passieren, daß in einem Markierungsrahmen

Dialogelemente enthalten wären, die Sie gar nicht markieren wollen.).
Um [*Umschalt*]-Klicken auszuführen, markieren Sie durch Klicken das erste Element, halten die [*Umschalttaste*] gedrückt und klicken auf die weiteren Dialogelemente, die Sie markieren wollen.

Um Dialogelemente dieser Gruppe von markierten Elementen hinzuzufügen oder um Elemente aus der Gruppe zu entfernen, können Sie auf die Dialogelemente, die Sie hinzufügen oder entfernen wollen, [*Umschalt*]-Klicken.

Mit *Bearbeiten/Alles markieren* können Sie alle Dialogelemente eines Fensters markieren. Resource Workshop zieht einen Rahmen um jedes Element und die ganze Gruppe von Dialogelementen wird mit einem Markierungsrahmen umgeben. Mit *Bearbeiten/Alles markieren* wird jedoch nicht der Rahmen des Dialogfensters ausgewählt.
Sie können auch mit *[Tab]* Dialogelemente markieren.

Mit Klicken auf das Werkzeug Reihenfolge bestimmen können Sie die Reihenfolge der Dialogelemente sehen.

- Ist ein Dialogelement ausgewählt, wird mit *[Tab]* das nächste Element in der Reihenfolge, wie sie ursprünglich eingegeben worden sind, markiert. Mit [*Umschalt*]-*[Tab]* verschieben Sie die Markierung zum vorhergehenden Element dieser Reihenfolge.
 In jedem Fall wird mit *[Tab]* der Fensterrahmen markiert, wenn Sie das letzte Element der Reihenfolge markiert hatten.
- Haben Sie mit dem Markierungsrahmen eine Gruppe von Dialogelementen markiert, verschieben Sie mit *[Tab]* die Markierung auf das Nachfolgeelement des zuletzt eingegebenen Elements der Gruppe. Die Gruppe wird aufgelöst.
 Enthält z.B. die Gruppe die Dialogelemente 4, 5, 9 und 10, würden Sie mit *[Tab]* das Element 11 markieren, sofern es dieses gibt.
- Haben Sie mit [*Umschalt*]-Klicken eine Gruppe zusammengestellt, so wird mit *[Tab]* das Nachfolgelement zum zuletzt [*Umschalt*]-geklickten Gruppenelement markiert.
 Haben Sie in der Reihenfolge 9, 10, 4 und 5 die Dialogelemente als Gruppe mit [*Umschalt*]-Klicken markiert, so wird mit *[Tab]* das Element 6 markiert.

Dialogelemente einem Dialogfenster hinzufügen

Der einfachste Weg, Dialogelemente Ihrem Fenster hinzuzufügen, ist:

1. Klicken Sie auf das gewünschte Element in der Werkzeugpalette. Ihr Mauszeiger zeigt durch seine Form den Typ des hinzuzufügenden Dialogelements an.
2. Klicken Sie auf die Position im Fenster, an der dieses Element plaziert werden soll.

Wenn Sie ein Element aus der Werkzeugpalette ausgesucht haben und dann doch nicht verwenden wollen, klicken Sie einfach auf den Auswahl-

zeiger. Der gewohnte Pfeil erscheint wieder und Sie können Elemente des Dialogfensters markieren. Mit *[Esc]* können Sie auch die Positionierung eines Elements abbrechen und zum Auswahlmodus zurückkehren.

Sie können auch das Steuermenü des Dialogeditors verwenden (in dessen Menüleiste, nicht an der Fenstergrenze), um weitere Dialogelemente hinzuzufügen.

1. Mit Ihrer Maus oder *[Alt]*+*[C]* öffnen Sie das Steuermenü.
2. Wählen Sie den Typ des gewünschten Dialogelements.
3. Klicken Sie auf die Position im Fenster, an der dieses Element plaziert werden soll.

Mehrere Kopien eines Dialogelements in ein Dialogfenster einfügen

Sie können mehrere Kopien desselben Dialogelements in Zeilen oder Spalten in Ihr Dialogfenster einfügen. Wenn Sie z.B. je vier Markierungsfelder in zwei Reihen in Ihr Dialogfenster einfügen wollen, können Sie jedes Markierungsfeld gesondert hinzufügen. Sie können aber auch viel einfacher vorgehen.

Sie fügen mehrere Kopien desselben Dialogelements wie folgt in Reihen oder Zeilen ausgerichtet ein:

1. Markieren Sie in Ihrem Dialogfenster das Dialogelement, das sie kopieren wollen.
2. Klicken Sie auf das Duplizieren-Werkzeug oder gehen Sie mit *Bearbeiten/Duplizieren* vor. Es erscheint das Dialogfenster Element duplizieren.
3. Geben Sie die Anzahl der Spalten und Zeilen sowie den Abstand in Dialogeinheiten dazwischen an.

 Wünschen Sie z.B. 8 Markierungsfelder in zwei Reihen, so geben Sie vier Zeilen und zwei Spalten an.
4. Klicken Sie auf *OK*. Die gewünschte Anzahl von Elementen erscheint perfekt ausgerichtet in Ihrem Dialogfenster.

➡ Wenn Sie nur ein Dialogelement markieren, hat das Duplizieren-Werkzeug die gleiche Wirkung wie *Bearbeiten/Duplizieren*. Haben Sie mehrere Elemente markiert, wirkt es wie *Ausrichten/Tabelle*.

Eigenschaften der Dialogelemente (.VBX Elemente)

Wenn Sie *Optionen/Eigenschaften anzeigen* einschalten, wird zum markierten Element das Eigenschaften Dialogfenster gezeigt, so daß Sie sich die Eigenschaften der Dialogelemente ansehen können, die Sie hinzufügen.

Eine .VBX Element kann nur über das Eigenschaften Dialogfenster bearbeitet werden. Sie sollten wissen, welche Eigenschaften Ihr Element aufweist, so daß Sie diese verändern können.

Wird das Dialogfenster nicht angezeigt, können Sie es durch Doppelklicken auf ein .VBX Element anzeigen lassen.

Dialogelemente verschieben

Sie können durch Ziehen ein markiertes Element an jede Stelle innerhalb des Markierungsrahmens bewegen. Auch in Gruppen markierte Elemente können auf gleiche Weise verschoben werden. Beim Ziehen bewegen sich die Elemente zusammen, wobei die Position relativ zueinander beibehalten wird.

Auch mit Hilfe der Tastatur kann ein Element bewegt werden. Mit [*Tab*] markieren Sie es. Mit den Pfeiltasten bewegen Sie es. Mit der [*Eingabetaste*] bestimmen Sie die endgültige Position. Wenn Sie statt der [*Eingabetaste*] [*Esc*] drücken, wird das Bewegen rückgängig gemacht.

Die Größe von Dialogelementen verändern

Sie können die Größe eines markierten Elements durch Ziehen am entsprechenden Rand oder der entsprechenden Ecke verschieben. Mit Tastatur und Maus können Sie eine »Feineinstellung« der Größenveränderung vornehmen.

1. Markieren Sie ein Element und bewegen Sie den Mauszeiger zum gewünschten Rand des Elements.
2. Erscheint der Mauszeiger als Doppelpfeil, halten Sie die linke Maustaste gedrückt.
3. Drücken Sie die entsprechende Pfeiltaste so oft, bis Sie den Mauszeiger um die gewünschte Anzahl von Dialogeinheiten bewegt haben. Jeder Pfeiltastendruck entspricht einer Dialogeinheit.

Das gleichzeitige Positionieren und Verändern der Größe von Dialogelementen

Mit dem Dialogfenster Größe der Elemente ändern können Sie gleichzeitig die Position und die Größe eines Dialogelements bestimmen.

1. Markieren Sie das Element.
2. Wählen Sie *Ausrichten/Größe* oder drücken Sie die *[Alt]*-Taste und doppelklicken Sie auf das Element. Das Fenster Größe der Elemente ändern wird angezeigt.
3. Um die Position der linken oberen Ecke des Elements zu bestimmen, geben Sie die X- und Y-Koordinaten in Dialogeinheiten ein. Mit den Koordinaten 0,0 wird das Element in die obere linke Ecke des Dialogfensters positioniert, direkt unter der Titelleiste.

Um die Breite und Höhe des Elements zu bestimmen, geben Sie diese als CX- und CY-Werte in Dialogeinheiten ein. Sehen Sie dazu Abb. 15.4.

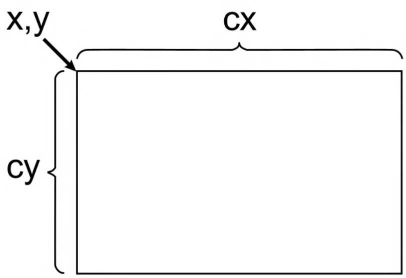


Abbildung 15.4: Dialogfenster-Koordinaten

Das Ausrichten von Dialogelementen mit Hilfe eines Gitters

Sie können sich ein Gitter im Dialogfenster anzeigen lassen und es benutzen, um Ihre Elemente auszurichten. So zeigen Sie ein Gitter an:

1. Mit *Ausrichten/Gitter* erscheint das Fenster Gitterattribute festlegen.
2. Geben Sie Höhe und Breite einer Gitterzelle in Dialogeinheiten ein.
3. Wählen Sie den Gittertyp. Es gibt dabei zwei Möglichkeiten:
 - Absolut. Verschiebt das Dialogelement zur nächstliegenden Gitterlinie.
 - Relativ. Verschiebt das Element nur in Vielfachen der Gitterzellenweite waagrecht oder in Vielfachen der Gitterzellenhöhe senkrecht. War das Element dabei nicht ursprünglich an einer Gitterlinie ausgerichtet, wird Ihnen dies bei Auswahl von *Relativ* auch nicht gelingen.

 Haben Sie z.B. ein 4x4 Gitter und ein Element auf der Position (1,1), können Sie das Element nur vier Einheiten in einer Richtung verschieben. Mögliche Koordinaten wären dann (5,5),(5,9),(9,5) usw.
4. Markieren Sie die Option Gitter zeigen und klicken Sie auf OK.

Dialogelemente bearbeiten

Um ein Element Ihres Dialogfensters zu bearbeiten, doppelklicken Sie auf das Element. Das Dialogfenster Eigenschaften zu diesem Element erscheint. Die Optionen im Fenster sind je nach gewähltem Element verschieden. Wenn Sie z.B. auf einen Schalter doppelklicken, erscheint das Dialogfenster Schaltereigenschaften.

Wollen Sie mit der Tastatur arbeiten, markieren Sie mit *[Tab]* das zu bearbeitende Element. Mit Drücken der [*Eingabetaste*] wird das Dialogfenster Eigenschaften zu diesem Element angezeigt.

Obwohl jeder Typ von Element sein eigenes Eigenschaftsfenster hat, haben die Fenster viele Optionen gemeinsam.

Tabelle 15.1: Optionen, die in allen Dialogfenstern Eigenschaften der Dialogelemente aufgeführt sind.

Option	Beschreibung
Titel	Hier können Sie den Titel eingeben, der mit Ihrem Element erscheinen soll. Verschiedene Elementtypen zeigen den Titel an verschiedenen Stellen an. So wird z.B. bei einem Gruppenfeld der Titel in der linken oberen Ecke angezeigt, bei einem Schalter innerhalb des Schaltfeldes. Nicht alle Elemente zeigen den Titel an. Ein Listenfeld z.B. zeigt den als Titel eingegebenen Text nicht an. Rechts von der Texteingabemöglichkeit für Ihren Titel können Sie Text oder Nummer markieren. Markieren Sie Text, wird der Titel in der .RC oder Quelldatei mit Anführungsstrichen umgeben. Wollen Sie keine Anführungsstriche, markieren Sie Nummer.
ID Wert	Hier können Sie einen einmaligen Bezeichner Ihrem Element zuweisen. ID-Werte können vom Typ short integer oder integer sein. Konventionsgemäß wird statischen Elementen, die während der Laufzeit Ihrer Anwendung nicht verändert werden, der Bezeichner -1 zugewiesen. Wenn Sie einen alphanumerischen Wert für die ID eingeben, wird zunächst geprüft, ob ein Bezeichner bereits als *#define* oder *Konstantendeklaration* besteht. Wenn dies nicht der Fall ist, werden Sie gefragt, ob Sie einen Bezeichner erzeugen wollen. In Kapitel 14 finden Sie nähere Informationen über Bezeichner.
Bildlaufleiste	Sie können eine waagrechte oder senkrechte Bildlaufleiste Ihrem Element beifügen.

Die meisten Elemente haben folgende Eigenschaften gemeinsam:

Tabelle 15.2: Attribute der Elemente

Attribut	Beschreibung
Tab	Mit [*Tab*] kann der Benutzer dieses Element erreichen.
Gruppe	Damit wird das erste Element der Gruppe bestimmt.
Nicht verfügbar	Stellt das Element grau dar, um anzuzeigen, daß es auf Eingaben des Benutzers nicht reagiert.
Sichtbar	Es wird bestimmt, ob das Element sichtbar sein wird, wenn das Dialogfenster zuerst aufgerufen wird. Voreinstellungsgemäß ist die Option markiert (WS_VISIBLE). Ist diese Option nicht markiert (NOT WS_VISIBLE), erscheint das Element zunächst nicht. Ihre Anwendung kann dieses Element durch Aufruf der Funktion *ShowWindow* während der Laufzeit anzeigen.
Rahmen	Es wird ein Rahmen um das Dialogelement angezeigt.

Jeder Typ eines Elements hat zudem Optionen, die spezifisch für diesen Typ sind. Diese Optionen werden in den Abschnitten beschrieben, die sich mit den einzelnen Elementtypen beschäftigen.

Hinzufügen von Titeln zu Dialogelementen

Neben dem Eingeben von Titeln in Eigenschaften Dialogfenstern können Sie Titel für das Element auch anders eingeben.

Um einem Elementeinen Titel zu geben, markieren sie es und wählen eine der folgenden Vorgehensweisen:

- Sie wählen *Optionen/Eigenschaften anzeigen* und geben den Titel ins oberste Textfeld des erscheinenden Fensters ein.
- Sie doppelklicken das Element (oder Dialogfenster) und geben dann einen Titel in das Titel-Eingabefeld ein.

Die Klasse eines Elements verändern

Wenn Sie mit benutzerdefinierten Dialogelementen arbeiten, könnten Sie das Dialogfenster Allgemeine Elementeigenschaften hilfreich finden. Es wird angezeigt, wenn Sie bei gedrückter [*Strg*]-Taste auf Ihr Element doppelklicken oder mit *Tab* das Element markieren und dann [*Strg*] und die [*Eingabetaste*] drücken.

Im Dialogfenster Allgemeine Elementeigenschaften können Sie die Klasse des Dialogelements verändern. Sie können auch den Titel, den ID-Wert und Stil definieren. Geben Sie etwas unter Info ein, wird Ihr Dialogfenster nicht mehr mit dem Microsoft Ressourcen-Compiler kompatibel sein.

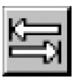
Bestimmen von Dialogelementen als Tabulatorpositionen

Die Werkzeuge Tabulator, Elementgruppen und Reihenfolge bestimmen haben Einfluß auf die Interaktion des Benutzers mit dem Dialogfenster über die Tastatur.

Bei Verwendung der Tastatur springen die Benutzer für gewöhnlich mit Hilfe der *Tab*-Taste von einem Element (oder einer Elementgruppe) zum nächsten. Bestimmte Elementtypen sind automatisch mit Drücken der *Tab*-Taste erreichbar, wenn Sie sie Ihrem Dialogfenster hinzufügen. Sie können einstellen, welche Elemente als Tabulatorposition definiert und damit durch Drücken der *Tab*-Taste erreichbar sind. Dadurch beeinflussen Sie, wie sich der zukünftige Benutzer mit der *Tab*-Taste in Ihrem Dialogfenster bewegen wird.

Sie können auf zweierlei Art Tabulatorpositionen verändern:

- Benutzen Sie das Tabulator-Werkzeug oder den Kommando Tabulator setzen
- Benutzen Sie das Eigenschaften Dialogfenster.



Um Tabulatoren zu setzen oder zu verändern, verwenden Sie das Tabulator Werkzeug oder den Tabulator setzen Kommando.

1. Mit *Optionen/Tabulator* setzen oder Klicken auf das Tabulatorwerkzeug verändert sich der Mauszeiger sein Aussehen und erscheint als Tabulator setzen-Symbol. Die als Tabulatorposition schon ausgewählten Dialogelemente werden im Resource Workshop mit einem grauen Rahmen umgeben.
2. Um eine Tabulatorposition zu setzen, klicken Sie auf eines der Dialogelemente, die nicht von einem grauen Rahmen umgeben sind. Um eine Tabulatorposition zu entfernen, klicken Sie auf eines der Dialogelemente, die von einem grauen Rahmen umgeben sind.
3. Sind Sie mit der Bearbeitung von Tabulatorpositionen fertig, klicken Sie auf das Markierungswerkzeug. Sie können dann mit dem Bearbeiten Ihres Dialogfensters fortfahren.

Sie können auch das Eigenschaften Dialogfenster zur Veränderung von Tabulatorpositionen nutzen:

1. Öffnen Sie das Eigenschaften Dialogfenster durch doppelklicken auf das Element oder indem Sie es markieren und dann die [*Eingabetaste*] drücken.

2. Markieren Sie Tab unter den Attributen, um das Element als Tabulatorposition zu definieren, oder machen Sie die Markierung rückgängig, um diese Einstellung abzuschalten.

Sie können Ihr Dialogfenster testen, um zu sehen, wie das Bewegen mit der *Tab*-Taste in Ihrem Dialogfenster vor sich geht.

Das Gruppieren zusammenhängender Dialogelemente



Sie können Gruppen von Elementen definieren. Der Benutzer kann sich mit Hilfe der Pfeiltasten in den Dialogelementen einer Gruppe bewegen.

Das Definieren von Gruppen geht im Grunde genauso vor sich wie das Bestimmen von Tabulatorpositionen. Sie benutzen dazu das Elementgruppen-Werkzeug oder das Kommando *Optionen/Gruppen* bestimmen. Durch einen grauen Rahmen zeigt Resource Workshop an, daß ein Element als Gruppe ausgezeichnet wurde. Sie können auch das Attribut Gruppe im Eigenschaftenfenster markieren.

Beachten Sie aber, daß Sie nur das *erste* Element der Gruppe bestimmen. Ausgehend von der Reihenfolge, in der Sie die Dialogelemente eingegeben haben, werden von Resource Workshop alle nachfolgenden (d.h. später hinzugefügten) Dialogelemente als Teil dieser Gruppe behandelt, bis wieder ein Element auftaucht, das als erstes Element einer weiteren Gruppe bestimmt wurde.

Sie müssen sich folgende wichtige Tatsachen über Gruppen vor Augen halten:

- Die Reihenfolge, in der Sie die Dialogelemente dem Dialogfenster hinzugefügt haben, ist wichtig.
- Wenn Sie eine »Gruppe« von Dialogelementen durch Ziehen eines Rahmens oder [*Umschalt*]-Klicken markieren, sind sie nicht eine Gruppe im Sinne dieses Abschnitts. Es handelt sich dann einfach um mehrere markierte Dialogelemente. Das Attribut Gruppe kann nur mit dem Elementgruppen-Werkzeug oder dem Kommando *Optionen/Gruppen* bestimmen oder mit dem Markierungsfeld Gruppe im Eigenschaften Dialogfenster vergeben werden.

Dialogelemente umordnen

Sie können die Reihenfolge bestimmen, in der ein Benutzer durch Drücken der *Tab*-Taste die Dialogelemente im Fenster erreicht. Wie in den vorhergehenden Abschnitten beschrieben, ist diese Reihenfolge beson-

ders wichtig, wenn Sie Gruppen von zusammengehörenden Elementen definiert haben.

Um die Reihenfolge zu bestimmen, in der ein Benutzer durch Drücken der *Tab*-Taste die Dialogelemente im Dialogfenster erreicht,

1. markieren Sie die Dialogelemente, deren Reihenfolge geändert werden soll.
 Soll die Reihenfolge aller Dialogelemente verändert werden, markieren Sie überhaupt kein Element.



2. Klicken Sie auf das Werkzeug Reihenfolge Bestimmen. Der Mauszeiger wird als Reihenfolge bestimmen-Symbol angezeigt. Jedes der Dialogelemente ist numeriert und zeigt so seinen Platz in der gesamten Reihenfolge an. Haben Sie im ersten Schritt nur einige Dialogelemente markiert, so sind nur deren Reihenfolgenummern zu sehen.
 Beachten Sie die Anzeige Nächstes am unteren Rand des Dialogeditors. Es gibt die Reihenfolgennummer für das nächste Element an, das Sie anklicken werden.
3. Klicken Sie die Dialogelemente an, die Sie neu numerieren wollen. Der Dialogeditor zeichnet einen grauen Rahmen um jedes Element, dem Sie durch Anklicken schon einen Reihenfolgenummer zugewiesen haben.
 Während Sie noch Nummern zuweisen, können Sie den letzten Schritt rückgängig machen, indem Sie auf das Element klicken, dem Sie zuletzt eine Nummer zugewiesen haben. Der Reihenfolgenzustand wird auf den vorhergehenden Zustand zurückversetzt. Sie können auf diese Weise beliebig weit zurückgehen, bis Sie die Ausgangsreihenfolge wiederhergestellt haben - Sie klicken einfach in umgekehrter Reihenfolge alle bisher angeklickten Elemente erneut an.
4. Mit Klicken auf das Markierungswerkzeug wird das Reihenfolge bestimmen beendet und Sie können weiter Ihr Dialogfenster bearbeiten.

Mit *Optionen/Reihenfolge* Bestimmen können Sie auch die Dialogelemente umordnen. Mit *Optionen/Elemente* kehren Sie zum Bearbeiten des Dialogfensters zurück. (Sie werden dieses Kommando nicht benötigen, wenn Sie alle markierten Dialogelemente anklicken.)

Ausrichten, Anordnen und Größe verändern von Dialogelementen

Wenn Sie schon Dialogelemente Ihren Fenster hinzugefügt haben, können Sie mit dem Dialogeditor Menü die Elemente ausrichten, in Spalten oder Zeilen anordnen und ihre Größe verändern.

Mehrere Dialogelemente ausrichten

Um Dialogelemente mit dem Kommando Ausrichten auszurichten,

1. markieren Sie die auszurichtenden Dialogelemente.
 Beachten Sie, daß der Markierungsrahmen alle markierten Dialogelemente umfaßt. Die einzelnen Elemente, die von den Ausrichtungsoptionen betroffen sein werden, sind grau umrahmt.
2. Mit *Ausrichten/Ausrichten* erscheint das Dialogfenster Elemente Ausrichten
3. Markieren Sie die gewünschten vertikalen und horizontalen Optionen zum Ausrichten. Mit Klicken auf *OK* werden die markierten Dialogelemente bewegt.

Tabelle 15.3: Ausrichtungsoptionen

Option	Beschreibung
Horizontale Ausrichtung	
Unverändert	Keine Veränderung in der waagrechten Ausrichtung.
Linke Seite	Richtet die Dialogelemente am linken Rand des Markierungsrahmens aus.
Zentriert	Richtet die Dialogelemente waagrecht am Zentrum des Markierungsrahmens aus.
Rechte Seite	Richtet die Dialogelemente am rechten Rand des Markierungsrahmens aus.
Gleicher Abstand	Bewegt die Dialogelemente waagrecht im Markierungsrahmen so, daß der Platz zwischen ihnen gleich groß wird.
Zentriert im Dialog	Bewegt den Markierungsrahmen waagrecht so, daß er im Zentrum des Dialogfensters steht. Die Position der Dialogelemente zueinander im Markierungsrahmen bleibt unverändert.
Vertikale Ausrichtung	
Unverändert	Keine Veränderung in der senkrechten Ausrichtung
Oberer Rand	Richtet die Dialogelemente am oberen Rand des Markierungsrahmens aus
Zentriert	Richtet die Dialogelemente senkrecht am Zentrum des Markierungsrahmens aus
Unterer Rand	Richtet die Dialogelemente am unteren Rand des Markierungsrahmens aus
Gleicher Abstand	Bewegt die Dialogelemente senkrecht im Markierungsrahmen so, daß der Platz zwischen ihnen gleich groß wird.
Zentriert im Dialog	Bewegt den Markierungsrahmen senkrecht so, daß er im Zentrum des Dialogfensters steht. Die Position der Dialogelemente zueinander im Markierungsrahmen bleibt unverändert.



Statt des Fensters Elemente Ausrichten können Sie auch Werkzeuge aus der Ausrichtungspalette benutzen. Markieren Sie die auszurichtenden Dialogelemente und wählen Sie dann in der Werkzeugpalette.

Die einzige Option, die auf diese Weise nicht erreichbar ist, ist die Option Gleicher Abstand. Natürlich können Sie dazu dann das Dialogfenster Elemente Ausrichten verwenden. Ebenso können Sie den Markierungsrahmen »ausdehnen«:

1. Markieren Sie die Dialogelemente, die Sie mit gleichem Abstand ausrichten wollen. Ein Markierungsrahmen umgibt diese Elemente.

2. Dehnen oder »strecken« Sie den Markierungsrahmen, indem Sie die [*Strg*]-Taste gedrückt halten und einen Rand des Rahmens in die Richtung ziehen, in die Sie gleichen Abstand zwischen den Dialogelementen herstellen wollen.
 - Um die Dialogelemente in horizontaler Richtung in gleichmäßigen Abständen anzuordnen, strecken Sie den rechten oder linken Rand des Markierungsrahmens.
 - Um die Dialogelemente in vertikaler Richtung in gleichmäßigen Abständen anzuordnen, strecken Sie den oberen oder unteren Rand des Markierungsrahmens.

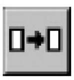
Wenn Sie am *Rand* des Dialogfensters ziehen, erscheint das Dialogfenster Elemente in Tabellenform ausrichten, so daß Sie die Dialogelemente in Spalten oder Zeilen anordnen können.

Dialogelemente in Zeilen oder Spalten anordnen

Das Kommando *Ausrichten/Tabelle* ordnet die Dialogelemente in Zeilen oder Spalten an, richtet sie senkrecht oder waagrecht aus und numeriert sie so um, daß sie sich in einer aufeinanderfolgenden Reihenfolge befinden.

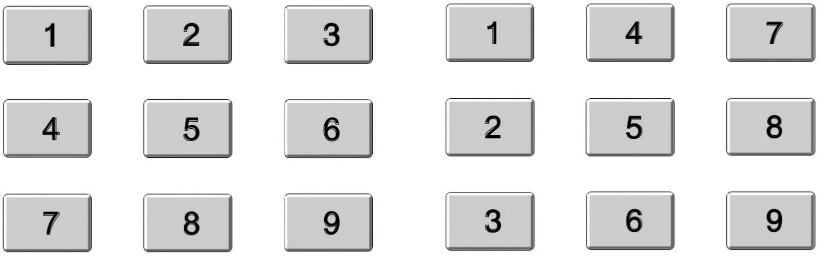
Um das Kommando zu benutzen,

1. markieren Sie die Dialogelemente, die Sie in Zeilen oder Spalten anordnen wollen.
 Beachten Sie, daß der Markierungsrahmen alle markierten Dialogelemente umfaßt. Die einzelnen Elemente, die von dem *Ausrichten/Tabelle* Kommando betroffen sein werden, sind grau umrahmt.
2. Wenn benötigt, vergrößern oder verkleinern Sie den Markierungsrahmen. Der Markierungsrahmen begrenzt das Gebiet, in dem die zu bildenden Zeilen oder Spalten eingefügt werden sollen. Machen Sie z.B. den Rahmen größer, werden die Dialogelemente zu den neuen Grenzen auseinander bewegt.



3. Klicken Sie auf das Duplizieren Werkzeug oder wählen Sie *Anordnen/Tabelle*.
4. Im Feld Tabellen-Layout geben Sie die gewünschte Anzahl der Zeilen und Spalten an.
5. Unter Reihenfolge markieren Sie, wie die Dialogelemente angeordnet werden sollen. In der folgenden Abbildung sehen Sie, wie die zwei Optionen neun Dialogelemente in drei Reihen anordnen würden.

Abbildung 15.5: Optionen für die Reihenfolge von Dialogelementen



6. Nach Auswahl der Optionen klicken Sie auf *OK*.

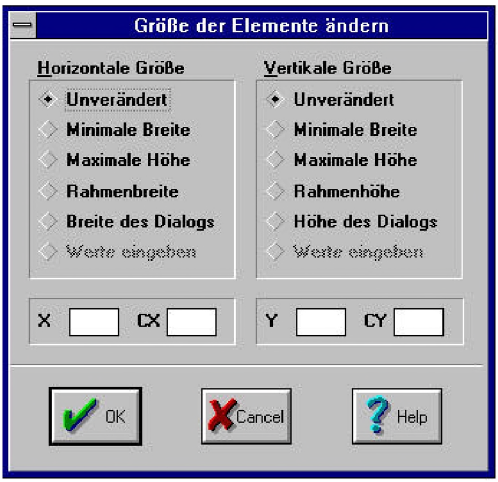
Für mehrere markierte Dialogelemente hat das Duplizieren Werkzeug die gleiche Wirkung wie *Ausrichten/Tabelle*. Ist nur ein Element ausgewählt, wirkt es wie *Bearbeiten/Duplizieren*.

Die Größe mehrerer Dialogelemente verändern

Sie können die Größe mehrerer markierter Dialogelemente mit den Optionen im Größe der Elemente ändern Dialogfenster verändern.

1. Markieren Sie die Dialogelemente, deren Größe Sie verändern wollen. Beachten Sie, daß der Markierungsrahmen alle markierten Dialogelemente umfaßt. Die einzelnen Elemente, die von den Optionen zur Größenveränderung betroffen sein werden, sind grau umrahmt.
2. Wählen Sie *Ausrichten/Größe*. Sie sehen folgendes Dialogfenster:

Abbildung 15.6: Dialogfenster Größe der Elemente ändern



3. Markieren Sie die gewünschten Optionen und klicken Sie auf *OK*, um die Änderungen durchzuführen.

Tabelle 15.4: Optionen zur Größenveränderung

Option	Beschreibung
Horizontale Größe	
Unverändert	Es gibt keine Größenveränderung in der Waagrechten.
Minimale Breite	Verkleinert die Breite der markierten Dialogelemente entsprechend dem schmalsten Dialogelement.
Maximale Höhe	Vergrößert die Ausdehnung der markierten Dialogelemente entsprechend dem breitesten Dialogelemente .
Rahmenbreite	Verändert die Ausdehnung der markierten Dialogelemente so, daß sie in der Breite genau in den Markierungsrahmen passen.
Breite des Dialogfenster	Verändert die Ausdehnung der markierten Dialogelemente so, daß sie so breit wie das Dialogfenster sind.
Vertikale Größe	
Unverändert	Es gibt keine Größenveränderung in der Senkrechten.
Minimale Breite	Verkleinert die Höhe der markierten Dialogelemente soweit, wie das kleinste der Dialogelemente hoch ist.
Maximale Höhe	Vergrößert die Ausdehnung der markierten Dialogelemente entsprechend dem höchsten Dialogelement.
Rahmenhöhe	Verändert die Ausdehnung der markierten Dialogelemente so, daß sie in der Höhe genau in den Markierungsrahmen passen.
Höhe des Dialogfenster	Verändert die Ausdehnung der markierten Dialogelemente so, daß sie so hoch wie das Dialogfenster sind.

Haben Sie die gewünschten Optionen markiert, klicken Sie auf *OK*.

Sie können die Auswirkungen der Optionen rückgängig machen durch *Bearbeiten/Rückgängig*, *[Alt]*+[*Rück*] oder mit dem Rückgängig Werkzeug.

Die Größe eines Dialogelements verändern

Wenn Sie nur ein einziges Element markieren und *Ausrichten/Größe* wählen, sind die folgenden Optionen verfügbar:

- Unverändert (Horizontal und Vertikal)
- Breite des Dialogfenster (Horizontal)
- Höhe des Dialogfenster (Vertikal)
- Werte eingeben (Horizontal und Vertikal)

Diese Optionen haben die Auswirkungen, die in Tabelle 15.4 beschrieben werden.

Mit der Option Werte Eingeben können Sie die Position und die Größe eines Dialogelements bestimmen. Um die Position der linken oberen Ecke des Elements zu bestimmen, geben Sie die X- und Y-Werte ein. Diese Werte bestimmen die Entfernung von der oberen linken Ecke des Dialogfensters (direkt unter der Titelleiste). Um die Breite und Höhe des Elements zu bestimmen, geben Sie diese als CX- und CY-Werte ein. Alle Werte werden in Dialogeinheiten angegeben.

Sie können die Auswirkungen jeglicher Bearbeitung im Dialogeditor durch *Bearbeiten/Rückgängig* oder mit dem Rückgängig Werkzeug zurück-

nehmen. Das Rückgängigmachen ist dabei sowohl bei Kommandos möglich, die mehrere Dialogelemente betreffen, als auch bei Kommandos, die nur ein Element betreffen.

Schaltelemente

Schaltelemente umfassen Schaltfelder, Aktionsschalter, Markierungsfelder und Gruppenfenster. Mit der Werkzeugpalette können Sie solche Schaltelemente Ihrem Dialogfenster hinzufügen. Zur Veränderung vorhandener Schaltelemente dient das Dialogfenster Schaltereigenschaften. Mit Doppelklicken auf ein zu bearbeitendes Schaltelement wird dieses Fenster angezeigt.

Sie können den Typ des Schaltelements ändern, indem Sie neue Optionen unter Schaltertyp im Dialogfenster Schaltereigenschaften markieren.

Tabelle 15.5: Schaltertypen

Schaltertyp	**Beschreibung**
Aktionsschalter	Der Schalter enthält Text. Mit Anklicken des Schalters durch den Benutzer wird eine BN_CLICKED Nachricht an das übergeordnete Fenster geschickt.
Voreingestellter Schalter	Das gleiche wie ein Schaltfeld, nur daß eine breite Umrahmung anzeigt, daß dieser Schalter aktiviert wird, wenn der Benutzer die [*Eingabetaste*] drückt.
Markierungsfeld	Ein rechteckiges Feld, zu dem links oder rechts ein Text gehören kann. Ist das Feld markiert, enthält es ein Kreuz. Diese Kennzeichnung des Feldes muß von der Anwendung vergeben oder zurückgenommen werden, wenn der Benutzer das Feld aktiviert.
Markierungsfeld (automatisch)	Ein Markierungsfeld wie eben, nur daß die Kennzeichnung von Windows vorgenommen wird.
3 Zustände	Ebenfalls ein Markierungsfeld, das einen dritten Zustand aufweisen kann. Das Feld wird grau gerastert, wenn der Zustand unbekannt oder unbestimmt ist. Die Anwendung muß das Feld kennzeichnen, die Kennzeichnung zurücknehmen oder es grau gerastert darstellen lassen.
3 Zustände (automatisch)	Wie das Schaltfeld mit den 3 Zuständen, nur daß Windows das Feld kennzeichnet, die Kennzeichnung zurücknimmt oder es grau gerastert darstellt.
Schaltfeld	Ein rundes Feld mit Text links oder rechts. Der Kreis wird mit einem massiven Punkt gefüllt, wenn das Feld aktiviert ist. Die Anwendung muß den Punkt setzen oder entfernen. Schaltfelder müssen in Gruppen auftreten. Gewöhnlich wird dem Benutzer so eine Menge von sich gegenseitig ausschließenden Optionen angezeigt. Bei Klicken auf das Feld wird eine BN_CLICKED Nachricht an das übergeordnete Fenster gesendet.
Schaltfeld (automatisch)	Wie das obige Schaltfeld, nur daß Windows den Punkt setzt oder entfernt.
Auswahlfeld	Ein rechteckiges Feld, in dem Schalter grafisch zusammengefaßt werden. Sie können einen Text angeben, der in der linken oberen Ecke des Feldes angezeigt wird.

Benutzerdefiniert	Dieser Typ von benutzerdefiniertem Schalter existiert aus Kompatibilitätsgründen zu Windows 2.0. Es wird empfohlen, diesen Typ unter Windows 3.0 nicht mehr zu verwenden und statt dessen untenstehenden selbstzeichnenden Schalter zu verwenden.
Selbstzeichnend	Das Schaltelement wird von der Anwendung und nicht von Windows ausgegeben. Wenn der Schalter gezeichnet werden soll, wird er die WM_DRAWITEM Nachricht an das übergeordnete Fenster senden.

Mit der Option Ausrichtung bestimmen Sie, ob der Text zu Schalt- oder Auswahlfeldern links oder rechts von diesen Feldern erscheint.

ID-Werte für Aktionsschalter

Windows definiert ID-Werte für Standard-Aktionsschalter, die in Dialogfenstern zum Verlassen verwendet werden. Sie können den ID-Namen (in Großbuchstaben) eingeben oder den ID-Wert aus Tabelle 15.6 entnehmen. Auf jedem Fall verändert im Unterschied zu Schaltern im Borland Stil (beschrieben in Anhang B) eine Änderung der Dialogelement-ID nicht den Text des Schalters. Um z.B. einen OK-Schalter zu erzeugen, müssen Sie *sowohl* den Namen der ID in IDOK *als auch* den Titel in OK ändern.

Tabelle 15.6: Die vordefinierten Windows Schalter

ID-Name	ID-Wert	Typ
IDOK	1	OK
IDCANCEL	2	Abbruch
IDABORT	3	Ende
IDRETRY	4	Wiederholen
IDIGNORE	5	Ignorieren
IDYES	6	Ja
IDNO	7	Nein

Bildlaufleisten

Fenster, Dialogfenster und Listenfelder benutzen Bildlaufleisten, um anzuzeigen, daß mehr Informationen vorhanden sind als augenblicklich angezeigt werden. Können z.B. in einem Dateilistenfeld nur 10 Dateien angezeigt werden, aber es sind insgesamt 20 Dateien vorhanden, so wird dies mit Bildlaufleisten angezeigt.

Eine Bildlaufleiste ist ein Rechteck mit Richtungspfeilen an beiden Enden. Das zwischen den Pfeilen liegende quadratische Symbol (Positionszeiger) zeigt in etwa die Lage des angezeigten Bereiches an. Liegt das Symbol z.B. auf der Hälfte der Bildlaufleiste, so sieht der Benutzer gerade die Dateien aus der Mitte der Liste.

Sie können waagrechte und senkrechte Bildlaufleisten an beliebigen Stellen in Ihr Dialogfenster einfügen.

Zum Positionieren der Bildlaufleiste benutzen Sie die Werkzeugpalette oder das Kommando *Element/Bildlaufleiste*. Zum Einstellen der Bildlaufleistenattribute verwenden Sie das Dialogfenster Eigenschaften der Bildlauf-

leiste, das nach Doppelklicken auf die zu verändernde Bildlaufleiste angezeigt wird.

Das Dialogfenster enthält neben den gemeinsamen Optionen (Tabelle 15.1) und Dialogelementattributen (15.2) Optionen, die die Bildlaufleiste innerhalb des Markierungsrahmens ausrichten.

Tabelle 15.7: Ausrichtungsoptionen

Option	Beschreibung
Keine	Die Bildlaufleiste füllt den gesamten Markierungsrahmen aus (Voreinstellung). Wenn Sie die Größe des Markierungsrahmens verändern, verändern Sie auch die Größe der Bildlaufleiste, und die Pfeile und Bildlaufleiste werden größer als gewöhnlich.
Oben Links	Eine waagrechte Bildlaufleiste wird am oberen Rand des Rahmens angezeigt und geht über die gesamte Breite des Rahmens. Eine weitere, senkrechte Bildlaufleiste wird am linken Rand des Markierungsrahmens angezeigt und geht über die gesamte Länge des Rahmens. Die Bildlaufleisten erscheinen immer in der Standardbreite.
Unten Rechts	Eine waagrechte Bildlaufleiste wird am unteren Rand des Rahmens angezeigt und geht über die gesamte Breite des Rahmens. Eine senkrechte Bildlaufleiste wird am rechten Rand des Markierungsrahmens angezeigt und geht über die gesamte Länge des Rahmens. Die Bildlaufleisten erscheinen immer in der Standardbreite.

Listenfelder

Ein Listenfeld ist ein Rechteck, das mehrere Textzeilen umfaßt. Gewöhnlich geht der Benutzer die Liste durch und markiert einen oder mehrere Einträge. Das Listenfeld sendet dann dem übergeordneten Fenster eine Nachricht, welche Einträge ausgewählt wurden.

Wenn die Liste der Einträge die Länge oder die Breite des Rechtecks überschreitet, kann das Listenfeld mit Bildlaufleisten versehen werden.

Neben den bereits beschriebenen gemeinsamen Optionen weist das Dialogfenster Eigenschaften der Liste die Optionen Selbstzeichnend und Listenfenster auf.

Mit *Selbstzeichnend* wird festgelegt, ob die Einträge in der Liste von der Liste selbst oder von der Anwendung ausgegeben werden. Wählen sie ein Attribut aus der folgenden Tabelle:

Tabelle 15.8: Die Optionen von Selbstzeichnend

Option	Beschreibung
Liste	Die Liste stellt die Einträge dar (Voreinstellung).
Anwendung	Die Anwendung zeichnet die Einträge der Liste als Antwort auf eine WM_DRAWITEM-Nachricht. Die Anwendung kann außerdem auf die Nachrichten WM_COMPAREITEM, WM_DELETEITEM und WM_MEASUREITEM antworten. Ein Listenfeld schickt nur dann eine WM_MEASUREITEM-Nachricht an die Anwendung, wenn die Einträge zum ersten Mal auszugeben sind. Hierbei wird auch die Höhe eines Eintrags festgelegt.

Variabel	Die Anwendung zeichnet die Einträge als Antwort auf eine WM_DRAWITEM-Nachricht. Die Anwendung kann außerdem auf die Nachrichten WM_COMPAREITEM, WM_DELETEITEM und WM_MEASUREITEM antworten. Die Liste schickt für jeden Eintrag eine WM_MEASUREITEM-Nachricht an die Anwendung; die Höhe der Einträge kann sich ändern.

Mit den Listenfenster-Optionen können Sie die Liste weiter definieren. Wählen sie eine oder mehrere Optionen der folgenden Tabelle:

Tabelle 15.9: Listenfenster-Optionen

Option	Beschreibung
Benachrichtigungen	Sendet eine EingabeNachricht an das übergeordnete Fenster, wenn der Anwender einen Listeneintrag anklickt (Voreinstellung).
Sortieren	Sortiert die Liste alphabetisch.
Mehrfachauswahl	Läßt den Anwender mehr als einen Eintrag gleichzeitig markieren. Der Anwender kann außerdem die Markierung einzelner Einträge an- und ausschalten.
Nicht neu zeichnen	Verhindert, daß die Liste bei einer Änderung neu gezeichnet wird.
Spaltenformat	Ordnet die Informationen spaltenweise in der Liste an. Die standardmäßige Breite beträgt 32 Dialogeinheiten oder 8 Zeichen. Sie sollten den Text mit Tabulatorzeichen (\x09) formatieren. (Wenn Sie die Spaltenbreite ändern möchten, sollte die Anwendung mit der LB_SETTABSTOPS-Nachricht eigene Tabulatorpositionen setzen.)
Höhenanpassung	Die Liste ändert ihre Höhe während der Laufzeit, so daß der Bereich groß genug ist, die Einträge vollständig zu zeigen (Voreinstellung). Wenn Höhenanpassung eingeschaltet ist und die Größe der Liste geändert werden muß, um die Einträge vollständig anzuzeigen, vermindert die Liste ihre Größe. Wenn zur Laufzeit beispielsweise drei Einträge vollständig angezeigt werden, aber der nächste nur noch teilweise in die Liste paßt, vermindert die Liste ihre Größe, so daß nur die drei Anträge gezeigt werden.
Mehrspaltig	Über einen horizontalen Bildlauf kann der Benutzer auf mehrere Spalten einer Liste zugreifen. Wenn sie diese Option einschalten, muß die Anwendung die Nachricht LB_SETCOLUMNWIDTH schicken, um die Breite aller Spalten in Pixeln einzustellen.
Eingabeweitergabe	Gibt die Eingaben des Benutzers an die Anwendung weiter.
Erweiterte Auswahl	Wenn diese Option bei Listen mit mehrfacher Auswahlmöglichkeit verwendet wird, ändert dieses Attribut die Funktionsweise der Mehrfachauswahl, so daß der Anwender mehr als einen Eintrag markieren kann.
Enthält Strings	Wenn Sie Anwendung oder Variabel gewählt haben, speichert die Liste den Text jedes Eintrags mit LB_INSERTSTRING oder LB_ADDSTRING. Die Liste kann Einträge mit der Nachricht LB_GETTEXT wieder abrufen.
Immer Bildlauf	Die Liste ist, unabhängig von der Zahl der Einträge, immer mit einer Bildlaufleiste ausgestattet. Diese Option entspricht der WINDOW.H-Konstanten LBS_DISABLENOSCROLL und ist nur unter Windows 3.1 verfügbar.

Eingabefelder

Mit einem Eingabefeld kann der Anwender Text über die Tastatur eingeben (wie zum Beispiel ein Dateiname im Dialogfenster Datei Öffnen)

Abbildung 15.7: Eingabefeld



Wenn Sie die Attribute des Eingabefelds festlegen möchten, klicken Sie es im Dialogfenster zweimal an. Es erscheint das Dialogfenster Eigenschaften des Eingabefeldes.

Das Dialogfenster Eigenschaften des Eingabefeldes enthält neben den schon beschriebenen gemeinsamen Optionen und den Elementattributen die folgenden Optionen:

- Ausrichtung
- Bildlaufleiste
- Automatischer Bildlauf
- Zeilen
- Groß- und Kleinschreibung
- Sonstige Optionen, einschließlich der Optionen für Windows 3.1

Folgende Tabelle beschreibt die Optionen des Dialogfensters Eigenschaften des Eingabefeldes.

Tabelle 15.10: Die Optionen des Dialogfensters Eigenschaften des Eingabefeldes

Option	Beschreibung
Ausrichtung	
Linksbündig	Richtet mehrzeiligen Text linksbündig aus (Voreinstellung).
Rechtsbündig	Richtet mehrzeiligen Text rechtsbündig aus.
Zentriert	Richtet mehrzeiligen Text mittig aus.
Bildlauf	
Horizontal	Das Eingabefeld hat eine horizontale Bildlaufleiste am unteren Fensterrand.
Vertikal	Das Eingabefeld hat eine vertikale Bildlaufleiste am rechten Fensterrand.
Automatischer Bildlauf	
Horizontal	Wenn die Eingabe den rechten Rand des Felds erreicht, wird der Text automatisch um zehn Zeichen nach links versetzt. Sobald der Benutzer die [*Eingabetaste*] drückt, wird der Text wieder an die Ausgangsposition zurückversetzt.
Vertikal	Wenn der Anwender in der letzten Zeile des Eingabefelds die [*Eingabetaste*] drückt, wird der Text eine ganze Seite nach oben versetzt. Wenn das Dialogelement beispielsweise fünf Zeilen hoch ist, wird der Text um fünf Zeilen nach oben versetzt, und der Mauszeiger erscheint wieder in der ersten Zeile. Sie müssen außerdem für das Eingabefeld Mehrere Zeilen anwählen, damit diese Option wirksam wird. **Anmerkung:** Das Drücken der [*Eingabetaste*] hat keine Wirkung, wenn sich Resource Workshop im Testmodus befindet. In Ihrer Anwendung wirkt es sich jedoch aus.

Groß-/Kleinschreibung	Zeigt den Text genau wie eingegeben an (Voreinstellung).
Großbuchstaben	Zeigt den gesamten Text in Großbuchstaben an, unabhängig davon, wie er eingegeben wurde.
Kleinbuchstaben	Zeigt den gesamten Text in Kleinbuchstaben an, unabhängig davon, wie er eingegeben wurde.
Zeilen	
Einzelne Zeile	Begrenzt das Eingabefeld auf eine einzige Zeile (Voreinstellung).
Mehrere Zeilen	Läßt den Anwender mehrzeiligen Text eingeben. (Um einen Bildlauf - durchführen zu können, muß die Option Automatischer Bildlauf/Vertikal gesetzt sein.)
Sonstige Optionen	
Paßwort	Wenn die Paßwort-Option eingeschaltet ist, wird die Anzeige der einzelnen Zeichen bei der Eingabe unterdrückt, und es erscheint stattdessen ein Stern. Diese Option ist hilfreich, Paßworte geheim zu halten.
OEM Konvertieren	Setzt den in dem Feld eingegebenen Text zuerst in den aktuellen OEM-Zeichensatz und wandelt den Text anschließend in ANSI. Diese Option ist für Dateieingabefelder hilfreich, da sie sicherstellt, daß jeder eingegebenen Dateiname in dem OEM-Zeichensatz übersetzbar ist; der OEM-Zeichensatz wird vom DOS-Dateisystem verwendet.
Markierung Beibehalten	Markiert den ausgewählten Text, selbst wenn dieses Feld nicht mehr über den Tastatur-Fokus verfügt. Als Beispiel nehmen wir an, daß der Anwender die Hälfte des Textes markiert aht, der in einem Eingabefeld angezeigt wird. Normalerweise ist der Text nicht mehr markiert, wenn der Benutzer zu einem anderen Dialogelement weitergeht. Wählen Sie Markierung Beibehalten, wenn der Anwender weiterhin markierten Text sehen soll, auch wenn er inzwischen mit anderen Dialogelementen arbeitet.

Die folgenden Optionen stehen nur bei Windows 3.1 zur Verfügung.

Tabelle 15.11: Eingabefeld-Optionen für Windows 3.1

Typ	**Beschreibung**
Nur lesen	Der Text enthält das Attribut Nur Lesen. Diese Option entspricht der Konstanten ES_READONLY in der Datei WINDOWS.H.
Rückgabe	Die [*Eingabetaste*] erzwingt einen Zeilenumbruch in einem Titeleingabefeld mit mehreren Zeilen. Wenn das Dialogelement nicht über den Tastatur-Fokus verfügt, wird durch Drücken der [*Eingabetaste*] der voreingestellte Aktionsschalter betätigt. Wenn die Option Rückgabe nicht gesetzt ist, wird ein Zeilenumbruch durch [*Strg*]+[*Eingabetaste*] erzeugt. Diese Option entspricht der Konstanten ES_WANTRETURN:

Statische Darstellung

Eine statische Darstellung zeigt Text oder Bilder, die der Anwender nicht verändern kann. Sie können mit statischen Darstellungen Teile Ihres Dialogfensters beschriften oder grafische Informationen darstellen.

Zu den statischen Darstellungen der Werkzeugpalette gehören

Die Werkzeuge für diese Elemente werden in Abbildung 15.3 dargestellt

- Statischer Text
- Rahmen
- Symbole
- Rechtecke

Sie können statische Darstellungen mit dem Dialogfenster Eigenschaften Statischer Elemente anpassen. Doppelklicken Sie auf die statische Darstellung, um das Dialogfenster Eigenschaften Statischer Elemente anzuzeigen.

Das Dialogfenster Eigenschaften Statischer Elemente enthält neben den schon beschriebenen gemeinsamen Optionen und den Attributen mehrere besondere Optionen für statische Darstellungen.

Im Auswahlfeld Keine Unterstreichung können Sie das Unterstreichen von Zeichen ausschalten. Wenn die statische Darstellung Text enthält, können Sie ein Zeichen unterstreichen, indem Sie ihm das &-Zeichen voranstellen. Schalten Sie Keine Unterstreichung ein, wenn Sie das Unterstreichen unterdrücken möchten.

Mit den Elementtyp-Optionen können Sie weiterhin definieren, welcher Art die statische Darstellung ist. Wählen Sie aus den folgenden Optionen:

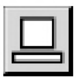
Tabelle 15.12: Die Elementtyp-Optionen

Option	Beschreibung
Linksbündiger Text	Zeigt den Text linksbündig innerhalb der Umrandung an (Voreinstellung). Wenn der Text mehrzeilig ist, werden Wörter, die über die Zeile hinausgehen, in die nächste Zeile umgebrochen. Generell wird die Schriftfarbe verwendet, die in der Systemsteuerung von Windows eingestellt ist.
Links kein Umbruch	Zeigt den Text linksbündig innerhalb der Umrandung an. Wenn der Text mehrzeilig ist, unterdrückt diese Option automatisch den Zeilenumbruch. Sie müssen die Zeile mit der [*Eingabetaste*] abschließen.
Zentrierter Text	Zeigt den Text mittig innerhalb der Umrandung an. Wenn der Text mehrzeilig ist, werden Wörter, die über die Zeile hinausgehen, in die nächste Zeile umgebrochen.
Rechtsbündiger Text	Zeigt den Text rechtsbündig innerhalb der Umrandung an. Wenn der Text mehrzeilig ist, werden Wörter, die über die Zeile hinausgehen, in die nächste Zeile umgebrochen. Mit Ausnahme des Typs Einfacher Text kann in allen statischen Text-Typen durch Eingabe von \T ein Tabulator, durch Eingabe von \R eine neue Zeile erzeugt werden.
Einfacher Text	Zeigt eine einzige Zeile mit linksbündigem Text an. Es können keine Tabulatoren gesetzt oder Zeilen umgebrochen werden. Einfacher Text - bearbeitet nicht die Nachricht WM_CTLCOLOR. Die Textfarbe entspricht der in der Systemsteuerung eingestellten Textfarbe; die Hintergrundfarbe entspricht der aktuellen Hintergrundfarbe von Windows.

Weißes Rechteck	Zeigt ein ausgefülltes Rechteck in der Hintergrundfarbe an, wie sie in der Systemsteuerung von Windows eingestellt ist. Weiß ist voreingestellt.
Graues Rechteck	Zeigt ein ausgefülltes Rechteck in der aktuellen Farbe des Desktops von Windows an, wie er in der Systemsteuerung von Windows eingestellt ist. Grau ist voreingestellt
Schwarzes Rechteck	Zeigt ein ausgefülltes Rechteck in der Rahmenfarbe an, wie sie in der Systemsteuerung von Windows eingestellt ist. Schwarz ist voreingestellt.
Weißer Rahmen	Zeigt einen leeren Rahmen in der Hintergrundfarbe an, wie sie in der Systemsteuerung von Windows eingestellt ist. Weiß ist voreingestellt.
Grauer Rahmen	Zeigt einen leeren Rahmen in der aktuellen Farbe des Desktops an. Grau ist voreingstellt.
Schwarzer Rahmen	Zeigt einen leeren Rahmen in der Rahmenfarbe an, wie sie in der Systemsteuerung von Windows eingestellt ist. Schwarz ist voreingestellt.
Symbol	Zeigt ein Symbol an. Wenn Sie das Symbol bearbeiten möchten, benutzen Sie das Bearbeiten-Symbol im Paint-Editor. **Anmerkung:** Wenn Sie Ihrem Dialogfenster einen Rahmen hinzufügen, scheint es eventuell so, als sei er mit der aktuellen Hintergrundfarbe des Fensters ausgefüllt. Wenn Sie in den Testmodus umschalten, sehen Sie den Rahmen, wie er zur Laufzeit aussieht.

Statische Symbole

Mit Resource Workshop können Sie in einem Dialogfenster Symbole anzeigen. Das gewünschte Symbol muß ein Teil des aktuellen Projekts sein (als eingebundene oder durch Verweis verbundene Ressource).



Um ein statisches Symbol in Ihr Dialogfenster zu plazieren, gehen Sie folgendermaßen vor:

1. Klicken Sie das Dialogelement Statisches Symbol in der Werkzeugpalette an und plazieren Sie es an der gewünschten Stelle im Dialogfenster.
2. Doppelklicken Sie das statische Symbol, um das Dialogfenster Eigenschaften statischen Textes zu öffnen.

Wenn das Titelfeld nicht den Ressourcennamen oder den Bezeichner des Symbols enthält, wird das Symbol nicht dargestellt.

3. Geben Sie im Titelfeld den Namen oder Bezeichner der Symbol-Ressource ein. Wenn Sie einen Namen eintragen, wählen Sie die Text-Option. Falls Sie einen Bezeichner eintragen - seinen Namen oder seine Nummer - müssen Sie die Nummer-Option wählen.
 Beispielsweise haben Sie in ihrer Projektdatei einen Bezeichner mit dem Namen GLOBE und der Nummer 1300. Jetzt müssen Sie Nummer wählen und können entweder GLOBE oder 1300 als Titel eingeben.
4. Wählen Sie *OK*. Das Symbol erscheint in Ihrem Dialogfenster.
5. Wenn Sie das Symbol bearbeiten möchten, doppelklicken Sie auf das Symbol, um das Dialogfenster Eigenschaften Statischer Elemente wieder anzuzeigen. Nun ist der Symbol Bearbeiten-Schalter verfügbar. Klicken Sie ihn an, um den Paint-Editor zu starten, mit dem Sie das Symbol bearbeiten können.

Kombinationsfenster

Ein Kombinationsfenster verbindet ein Listenfenster (aus dem der Anwender Einträge auswählen kann) entweder mit einer statischen Darstellung (einem Text, den der Benutzer nicht verändern kann) oder mit einem Eingabefeld (einem Bereich, in dem der Anwender Eingaben machen kann).

Abbildung 15.8: Kombinationsfenster des Dialogfensters Datei Öffnen



Sie können ein Kombinationsfenster mit dem Dialogfenster Eigenschaften des Kombinationsfensters anpassen. Doppelklicken Sie im Dialogfenster das Kombinationsfenster, und es erscheint das Dialogfenster Eigenschaften des Kombinationsfensters.

Mit drei Typ-Optionen können Sie die Art des Kombinationsfensters definieren.

Tabelle 15.13: Die Typ-Optionen des Kombinationsfensters

Option	Beschreibung
Einfach	Die Liste ist immer sichtbar und zeigt die Listeneinträge; der Anwender kann die Einträge bearbeiten (Voreinstellung).
Aufklappbar	Beim ersten Anzeigen des Dialogfensters besteht das Kombinationsfeld aus einer einzigen Textzeile, die bearbeitet werden kann. Durch Anklicken des nach unten gerichteten Pfeils wird die Liste geöffnet. Alle Listeneinträge können jetzt bearbeitet werden.
Nicht Editierbar	Arbeitet ähnlich wie aufklappbar, außer daß die Liste statisch ist. Der Anwender kann aus der Liste wählen, aber keine Änderungen in der Liste vornehmen.

Mit den Optionen Durch Anwendung legen Sie fest, ob die Liste vom Listen-Dialogelement oder von der Anwendung gezeichnet werden soll.

Tabelle 15.14: Selbstzeichnend-Optionen

Option	Beschreibung
Nein	Die Liste wird durch das Dialogelement Listenfeld selbst gezeichnet (Voreinstellung).
Anwendung	Die Anwendung zeichnet die Liste als Antwort auf eine WM_DRAWITEM-Nachricht. Die Anwendung kann außerdem auf die Nachrichten WM_COMPAREITEM, WM_DELETEITEM und WM_MEASUREITEM antworten. Die Liste schickt nur dann eine WM_MEASUREITEM-Nachricht an die Anwendung, wenn sie zum ersten Mal gezeichnet wird. Dabei wird gleichzeitig die Größe des Listenfensters festgelegt.
Variabel	Die Anwendung zeichnet die Liste als Antwort auf eine WM_DRAWITEM-Nachricht. Die Anwendung kann außerdem auf die Nachrichten WM_COMPAREITEM, WM_DELETEITEM und WM_MEASUREITEM antworten. Die Liste schickt für jeden Eintrag eine WM_Measureitem-Nachricht an die Anwendung; die Größe des Listenfensters kann sich deshalb ändern.
Enthält Strings	Wenn Sie Anwendung oder Variabel gewählt haben, speichert die Liste den Text jedes Eintrags über die Nachricht LB_SETTEXT. Die Liste kann Einträge mit der Nachricht LB_GETTEXT wieder abrufen.

Das Dialogfenster Eigenschaften des Kombinationsfensters enthält neben den beschriebenen gemeinsamen Optionen und den Attributen einige besondere Optionen für Kombinationsfenster.

Tabelle 15.15: Die Auswahlmöglichkeiten für Attribute des Kombinationsfensters

Option	Beschreibung
Vert. Bildlaufleiste	Die Liste erhält eine vertikale Bildlaufleiste.
Sortiert	Sortiert die Einträge automatisch alphabetisch.
Höhenanpassung	Die Liste kann ihre Höhe während der Laufzeit ändern, so daß alle Einträge vollständig gezeigt werden (Voreinstellung). Wenn Sie die Höhe der Liste genau einstellen müssen, können Sie diese Option ausschalten.
OEM-Konvertierung	Setzt den vom Anwender eingegebenen Text zuerst in den aktuellen OEM-Zeichensatz um und wandelt ihn anschließend in ANSI-Format um. Diese Option ist für Dateieingabe-Felder hilfreich, da sie sicherstellt, daß jeder eingegebene Dateiname in den OEM-Zeichensatz übersetzbar ist. Der OEM-Zeichensatz wird vom DOS-Dateisystem verwendet.
Horiz. Bildlauf	Versetzt den Text automatisch nach links, wenn er die Breite des Eingabefelds überschreitet.
Immer horiz. Bildlauf	(Nur bei Windows 3.1) Unabhängig von der Anzahl der Symbole wird immer eine vertikale Bildlaufleiste angezeigt. In der Datei WINDOWS.H ist die entsprechende Konstante dafür CBS_DISABLENOSCROLL.

Benutzerdefinierte Elemente

Wenn Sie ein Dialogelement verwenden möchten, das keinem der vordefinierten Typen in Windows entspricht, können Sie ein *benutzerdefiniertes Element* verwenden. Die vordefinierten Dialogelemente, die bisher in diesem Kapitel besprochen wurden, sowie Listen, Bildlaufleisten und Schalter, sind Standardelemente. Sie wurden von Microsoft entwickelt und sind Bestandteile von Windows. Im Gegensatz dazu ist jede andere Klasse von Elementen, die Sie Ihren Dialogfenstern hinzufügen möchten, ein benutzerdefiniertes Dialogelement

Es gibt zwei Arten von benutzerdefinierten Dialogelementen: die installierbaren und die anwendungsspezifischen. Installierbare benutzerdefinierte Elemente müssen in einer dynamischen Linkbibliothek vorliegen. Anwendungsspezifische benutzerdefinierte Elemente werden in der Anwendung selbst implementiert. Im Resource Workshop werden sie als graues Rechteck oder als leerer Rahmen dargestellt.

Eigene benutzerdefinierte Dialogelemente erstellen

Wenn Sie eigene benutzerdefinierte Dialogelemente erstellen möchten, müssen Sie sie entwerfen und in dynamischen Linkbibliotheken (DLL) speichern. Nähere Einzelheiten über die Erstellung von benutzerdefinierten Dialogelementen finden Sie in der Online-Datei CUSTCNTL.RW, die bei der Installation von Resource Workshop auf die Festplatte kopiert wurde.

Bibliothek mit benutzerdefinierten Elementen einrichten (.DLL oder .VBX)

Benutzerdefinierte Dialogelemente werden in dynamischen Linkbibliotheken gespeichert (DLL). Wenn Sie Ihrem Dialogfenster benutzerdefinierte Elemente hinzufügen möchten, müssen Sie die entsprechende dynamische Linkbibliothek installieren. Danach stehen die benutzerdefinierten Elemente in dieser dynamischen Linkbibliothek wie alle anderen Windows Dialogelemente zur Verfügung.

Wenn Sie eine dynamische Linkbibliothek mit benutzerdefinierten Dialogelementen installieren möchten, führen Sie folgende Schritte aus:

1. Wählen Sie im Dialogeditor *Optionen/Bibliothek Installieren*. Das Dialogfenster Neue Element-Bibliothek Installieren erscheint.
2. Geben Sie die Datei mit der dynamischen Linkbibliothek ein.
3. Klicken Sie auf *OK*.

Jetzt stehen Ihnen die Dialogelemente in dieser dynamischen Linkbibliothek zur Verfügung und Sie können sie den Dialogfenstern hinzufügen.

Um bestimmte benutzerdefinierte Dialogelemente (wie z.B. .VBX-Elemente) zu bearbeiten, zeigen Sie das Dialogfenster Eigenschaften an, indem Sie auf das Dialogelement klicken oder *Optionen/Eigenschaften Zeigen* wählen.

Benutzerdefinierte Dialogelemente anzeigen

Bevor Sie benutzerdefinierte Dialogelemente zu Ihren Dialogfenstern hinzufügen, müssen Sie entscheiden, wie sie im Dialogeditor angezeigt werden sollen. Wählen Sie *Optionen/Vorgaben* zur Prüfung, ob Ihre Dialogfenster im Entwurf- oder im Normal-Modus dargestellt werden. Wenn einer dieser beiden Modi eingeschaltet ist, können Sie die Option Benutzerdefinierte Elemente als Rahmen zeichnen ausschalten, damit Sie von Ihren benutzerdefinierten Dialogelementen mehr als nur einen Rahmen sehen.

Ein benutzerdefiniertes Dialogelement hinzufügen

Haben Sie erst die DLL Datei mit den benutzerdefinierten Dialogelementen installiert, können Sie jedes dieser Dialogelemente in Ihr Dialogfenster einfügen.
Wenn die benutzerdefinierten Dialogelemente vom Resource Workshop erkannt werden, erscheinen deren Symbole in der rechten Spalte der Werkzeugpalette und stehen sofort zur Verfügung.

Wenn Sie eine Bibliothek mit benutzerdefinierten Dialogelementen installiert haben,die nicht vom Ressource Workshop erkannt werden, müssen Sie auf folgende Weise vorgehen:

1. Klicken Sie das Symbol für Benutzerdefiniertes Element an oder wählen Sie *Element/Benutzerdefiniert*. Es erscheint das Dialogfenster Neues Benutzerdefiniertes Element.

2. Wählen Sie aus der mit dem Eingabefeld Klasse verbundenen Liste das gewünschte benutzerdefinierte Dialogelement. Ressource Workshop zeigt ein Beispiel des gewählten benutzerdefinierten Elements in der Mitte des Dialogfensters.
3. Nachdem Sie das gewünschte benutzerdefinierte Element markiert haben, klicken Sie auf *OK*. Aus dem Mauszeiger wird ein Fadenkreuz, woraus Sie ersehen können, daß Sie jetz das benutzerdefinierte Element plazieren können.
4. Klicken Sie an der Stelle des Dialogfensters, an der das benutzerdefinierte Element eingefügt werden soll.

Ein Dialogfenster testen



Um die Auswirkung der durchgeführten Änderungen zu begutachten und das Dialogfenster zu testen, klicken Sie das Test Werkzeug an oder wählen Sie *Optionen/Dialogfenster* testen.

Jetzt können Sie mittels [*Tab*] und [*Pfeiltasten*] feststellen, in welcher Reihenfolge die Elemente des Dialogfensters selektiert werden. Oder Sie geben Text ein und beobachten, wie der Text in einem Eingabefeld versetzt wird.

Beim Testen eines Dialogfensters wird in der Statuszeile des Dialogeditors die Meldung Test angezeigt.

Um den Test Modus zu verlassen, haben Sie folgende Möglichkeiten:

- Klicken Sie entweder auf den Schalter *OK* oder *Abbruch*.
- Wählen Sie noch einmal *Optionen/Dialogfenster* testen.
- Drücken Sie die [*Eingabetaste*].
- Doppelklicken Sie auf das Pfeilsymbol. Mit dem ersten Klicken wird der Dialogeditor statt des Dialogfensters aktiviert.

Zwei Dialogfenster anzeigen

Um gleichzeitig zwei Dialogfenster zu betrachten oder zu vergleichen, führen Sie folgende Schritte aus:

1. Doppelklicken Sie im Projektfenster auf eines der Dialogfenster, die Sie gleichzeitig darstellen möchten. Der Dialogeditor wird gestartet und zeigt dieses Dialogfenster.
2. Klicken Sie das Testwerkzeug an oder wählen Sie *Optionen/Dialogfenster* Testen.

3. Doppelklicken Sie auf das Verkleinerungssymbol des Dialogeditors. (Mit dem ersten Klicken wird der Dialogeditor statt des Dialogfensters aktiviert.) Das Dialogfenster ist jetzt ein nicht modales Dialogfenster, das Sie beliebig hin- und herbewegen können.
4. Kehren Sie zum Projektfenster zurück und klicken Sie das zweite Dialogfenster an, das Sie betrachten möchten. Ein zweiter Dialogeditor wird gestartet.
5. Klicken Sie wieder das Test-Werkzeug an oder wählen Sie im zweiten Dialogeditor *Optionen/Dialogfenster testen*.
6. Doppelklicken Sie auf das Verkleinerungssymbol des zweiten Dialogeditors.

Jetzt haben Sie zwei bewegliche Dialogfenster, die Sie nebeneinander positionieren können.

Zum Verlassen des Testmodus klicken Sie entweder auf *OK* oder *Abbruch*, oder doppelklicken Sie auf das Systemmenüfeld des Dialogfensters.

Dialogeditor anpassen

Bei Ressource Workshop können Sie die Arbeitsweise einiger Teile des Dialogeditors einstellen. Wählen Sie *Optionen/Vorgaben*, um das Dialogfenster Vorgaben anzuzeigen.

- Koordinatenbezug bestimmt die Maßeinheit bei der Anzeige von Informationen in der Statuszeile.
 - Dialog. Verwendet die Dialogeinheit als Maßeinheit. In Dialogeinheiten entspricht y 1/8 der Schrifthöhe und x 1/4 der Schriftbreite.
 - Bildschirm. Verwendet Pixel als Maßeinheit der Statuszeile.
- Mit der Option Markierungsrahmen können Sie die Erscheinungsform des Rahmens ändern, mit dem Dialogelemente zur Markierung umrandet werden.
 - Fetter Rahmen. Der Auswahlrahmen erscheint als dickes Rechteck um die gewählten Dialogelemente (Voreinstellung).

Ziehmarken. Die Auswahlrahmen um die gewählten Elemente werden mit Ziehmarken (kleinen Rechtecken) an den Ecken und den Mittelpunkten der Seiten versehen.

- Die Optionen Darstellung bestimmen, wie die Elemente des Dialogfensters im Dialogeditor dargestellt werden.

- Entwurf. Zeichnet Dialogelemente als Rechtecke mit der dazugehörigen Element-ID. Mit diesem Anzeigetyp ist sehr leicht die genaue Größe der Rechtecke festzustellen.
- Normal. Ahmt die Darstellung der Dialogelemente nach. Diese Option ist die schnellste Anzeigeart. Benutzerdefinierte Elemente werden aber nur als graue Rechtecke angezeigt.
- WYSIWYG (Voreinstellung). Mit dieser Option erstellt Ressource Workshop das Dialogfenster und die abhängigen Fenster für die Dialogelemente mit selbstzeichnenden Elementen. Diese Option ist langsamer, dafür aber die genaueste. Installierbare benutzerdefinierte Elemente zeichnen sich selbst.

■ Die Auswahloptionen legen fest, wie sich die Auswahl von Dialogelementen verhält. Wenn Sie mit eng beieinanderliegenden Dialogelementen arbeiten, könnten Sie diese Optionen markieren, um mit größerer Genauigkeit zu arbeiten. Außerdem können Sie diese Optionen verwenden, um die versehentliche Auswahl von Elementen zu vermeiden.

- Auswahl in Rahmennähe. Wenn diese Option eingeschaltet ist, müssen Sie genau auf den Rahmen eines Elements klicken, um das Element zu markieren. Wenn diese Option ausgeschaltet ist, wird das Element schon markiert, wenn Sie in seine Nähe klicken.

Auswahl durch umschließendes Rechteck. Wenn diese Option eingeschaltet ist, müssen Sie die Dialogelemente, die Sie mit dem Markierungsrahmen markieren möchten, vollständig umfassen, weil sie sonst nicht markiert werden. Wenn diese Option ausgeschaltet ist und der Markierungsrahmen nur einen Teil des Elements enthält, wird das Element trotzdem markiert.

Für weitere Informationen über die Ressourcen Skript Sprache können Sie das Hilfesystem nutzen.

In der Ressourcen Skript Sprache gibt es für jede Art von Dialogelementen eine eigene »Grammatikää. Für zentriert darzustellenden statischen Text wird beispielsweise die Anweisung CTEXT verwendet. Mit der CONTROL-Anweisung können Sie aber alle Arten von Dialogelementen angeben. Wählen Sie die Option Nur CONTROL-Anweisungen erzeugen, wenn Ressource Workshop in Ihren Ressourcen Skripts nur CONTROL Anweisungen anstelle der speziellen Anweisungen für die einzelnen Dialogelemente verwenden soll.

Das Markierungsfeld Benutzerdefinierte Dialogelemente als Rahmen ist nur verfügbar, wenn der Zeichnungstyp auf Normal eingestellt ist. Ist die Option markiert, werden benutzerdefinierte Dialogelemente als leere rechteckige Rahmen dargestellt. Ist die Option ausgeschaltet, werden benutzerdefinierte Dialogelemente als graugefärbte rechteckige Flächen dargestellt, wobei ihr Text (falls vorhanden) auf einem weißen Rechteck dar-

gestellt wird. Benutzerdefinierte Dialogelemente als leere Rahmen zu zeichnen, kann das Zeichnen des Dialogfensters erheblich beschleunigen.

Markieren Sie Ctl3d.dll, benutzt der Dialogeditor für Dialogelemente wie Markierungs- oder Auswahlfelder 3-D Effekte im Windows Stil.

K A P I T E L 16

Erzeugen von Menüs

Menüs sind Kommandolisten, aus denen der Benutzer ein Kommando wählen kann. Anwendungen, die Menüs verwenden, ersparen es dem Benutzer, sich eine Kommandosyntax merken zu müssen. Daher sind Anwendungen mit Menüs einfacher zu bedienen.

Die meisten Windows-Anwendungen haben eine Menüleiste am Oberrand ihres Bildschirms (bzw. Bildschirmausschnitts). Im Menüleiste befinden sich die Namen der Menüs für diese Anwendung. Jedes Menü enthält einen Kommandosatz. Die meisten Windows-Programme haben z.B. ein Datei-Menü, welches Kommandos für das Erzeugen, Öffnen, Speichern und Drucken von Dateien enthält.

Mit dem Menüeditor des Resource Workshop lassen sich leicht Menüs für Ihre Anwendung erzeugen und gestalten. Die Arbeit mit Menüs geschieht in vier grundlegenden Schritten.

1. Aufruf des Menüeditors. Sobald Sie eine neue Menüressource erzeugen, stellt Ihnen der Menüeditor eine Menüvorlage zum Bearbeiten zur Verfügung. Wenn Sie eine vorhandene Menüressource bearbeiten wollen, wird sie im Menüeditor angezeigt.
2. Das Menü wird erzeugt oder bearbeitet.
3. Das Menü wird getestet.
4. Das Menü wird gesichert.

Menü-Terminologie

Folgende Begriffe werden benutzt, um die Bestandteile einer Menüressource zu beschreiben:

- *Popup-Kommandos*. Diese Kommandos bewirken, daß Menüs angezeigt werden. Popup-Kommandos werden in der Menüleiste verwendet, wie z.B. die Standard-Windows-Datei und Hilfe-Menünamen. Sie können auch innerhalb von Menüs verwendet werden, wo sie die Anzeige eines weiteren Menüs verursachen können (Kaskaden-Menüs).
- *Popup-Menüs*. Dies sind rechteckige Felder, die Kommandolisten anzeigen. Aus den angezeigten Befehlen kann der Benutzer einen wählen. Popup-Menüs gibt es in zwei Erscheinungsformen:

Dropdown-Menüs. Diese werden in der Menüleiste oder innerhalb eines Menüs geöffnet. Sie sind mit einem Popup-Befehl verbunden und werden immer unter dessen Namen dargestellt. Zum Beispiel wird das Popup-Menü Datei direkt unterhalb des Kommandos Datei in der Menüleiste angezeigt.

Ad hoc-Menüs. Diese können überall im Fenster der Anwendung erscheinen. Ihre Position wird mit der Funktion *TrackPopupMenu* verwaltet.

Weitere Informationen über die Funktion *TrackPopupMenu* finden Sie im *Compiler Handbuch*.

- *Menüelemente*. Dies sind die Befehle, die in einem Menü erscheinen, wie z.B. Öffnen, Speichern oder Drucken.
- Separatoren. Dies sind Linien, mit deren Hilfe die Kommandos eines Menüs in Gruppen gegliedert werden können. Die Separatoren bewirken also nichts, außer daß die Menüs übersichtlicher und damit leichter benutzbar werden. Separatoren können nicht bearbeitet werden.

Aufruf des Menüeditors

In den beiden folgenden Abschnitten wird beschrieben, wie Sie den Menüeditor aufrufen, um ein Menü zu erzeugen oder zu bearbeiten.

Neues Menü erzeugen

Um ein neues Menü zu erzeugen

1. Stellen Sie sicher, daß das Projekt geöffnet ist, welchem Sie ein Menü hinzufügen wollen. Mit *Datei/Neues Projekt* erzeugen Sie ein neues Projekt, oder mit *Datei/Projekt öffnen* öffnen Sie ein bestehendes Projekt.

Kapitel 14 zeigt, wie ein neues Projekt geöffnet wird.

2. Ist das Projekt geöffnet, erzeugen Sie mit *Ressource/Neu* eine neue Ressource für das Projekt. Das Dialogfenster Neue Ressource erscheint.
3. Wählen Sie in der Ressourcentypen-Liste des Dialogfensters den Typ MENU. Doppelklicken Sie MENU, oder klicken Sie auf MENU und dann auf OK.

Der Menüeditor wird mit einem Standardmenü angezeigt, das Sie dann bearbeiten können.

Ein bestehendes Menü bearbeiten

Um ein bestehendes Menü zu bearbeiten, öffnen Sie das Projekt mit dem Menü und führen einen der folgenden beiden Schritte aus:

- Doppelklicken Sie auf den Menünamen im Projektfenster.
- Markieren Sie den Menünamen, und wählen Sie *Ressource/Bearbeiten*.

Der Menüeditor erscheint und hat schon das gewünschte Menü geladen.

Die Arbeitsfläche des Editors

Die Arbeitsfläche des Menüeditors ist in drei Teile gegliedert: Der Script-Ausschnitt zeigt das Menü wie ein Ressourcen-Script an. Zudem gibt es noch das Testmenü und den Dialogfeldausschnitt, der Ihnen die Bearbeitung der aktuell hervorgehobenen Zeile im Script-Ausschnitt ermöglicht. Mit dem Kommando Ansicht kann die Anordnung der Ausschnitte verändert werden.

Der Dialogfeldausschnitt

Im Dialogfeldausschnitt können Sie Popup-Kommandos und Menüelemente bearbeiten, ID-Werte zuweisen und Eigenschaften der Menüs und Menüelemente bestimmen. Sie können für jedes Kommando Tastenkürzel definieren. Die gerade bearbeitete Zeile wird im Script-Ausschnitt hervorgehoben dargestellt.

Der Testmenü-Ausschnitt

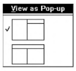
Im Testmenü-Ausschnitt wird Ihr Menü angezeigt, und Sie können es testen.

Das Popup-Menü für das Popup-Kommando umfaßt in der Voreinstellung standardmäßig ein Kommandoelement. Der Menüeditor bringt dieses Testmenü automatisch auf den neuesten Stand, wenn Sie Änderungen im Script-Ausschnitt vornehmen.

Mit dem Ansicht-Menü können Sie beeinflussen, wie das Testmenü dargestellt wird. Das gilt sowohl für die Menüleiste als auch für das Verhältnis zu den anderen Fenstern.

Tabelle 16.1: Optionen des Menüs Ansicht

Menüauswahl	Beschreibung
Darstellung als Popup-Menü	Bestimmt, ob die Popup-Befehle im Testmenü auf einer Menüleiste oder in einem Popup-Menü dargestellt werden. In der Voreinstellung ist diese Option ausgeschaltet, und die Popup-Befehle im Testmenü sind in der Menüleiste dargestellt.



	Lassen Sie die Option ausgeschaltet, wenn die Menüressource die gesamte Menüstruktur Ihrer Anwendung darstellt und Sie sehen möchten, wie sich die Menüstruktur dem Benutzer darstellt. Wenn Sie an einem Adhoc-Menü arbeiten, schalten Sie die Optionen durch Markieren ein. Sie können dann das Menü so sehen, wie es später tatsächlich erscheinen wird
Die erste Grafik	Die Abbildung repräsentiert die Standardeinstellung für die Ausschnitte. In der Standardeinstellung ist der Testmenü-Ausschnitt über dem Script-Ausschnitt rechts neben dem Dialogfeldausschnitt.
Die zweite Grafik	Wenn Sie diese auswählen, erscheint der Testmenü-Ausschnitt an der oberen Ebene des Editorfensters in Form einer normalen Menüleiste.

Der Script-Ausschnitt

Im Script-Ausschnitt werden die Popup-Kommandos, Menübestandteile und Separatoren des neuen Menüs in Pseudocode angezeigt. In der ersten Zeile finden Sie den Menünamen, die übrigen Zeilen sind Befehle zur Definition von Popup-Menüs und Menüelementen. Klicken Sie mit der rechten Maustaste in diesem Fenster, so wird ein Schnellmenü angezeigt (oder drücken Sie [*Alt*]+[*F10*] im Ausschnitt).

Der Pseudocode im Script-Ausschnitt ist eine vereinfachte Schreibweise, die es Ihnen ermöglichen soll, unkompliziert mit der Menüstruktur zu arbeiten. Beim Bearbeiten des Ressourcen-Scripts können Sie sich den kompletten Pseudocode und die Parameter für seine Kommandos ansehen.

Die Bearbeitung des Menüs erfolgt im Dialogfeldausschnitt. Mit [*F6*] können Sie zwischen dem Script-Ausschnitt und dem Dialogfeldausschnitt hin- und herschalten.

Um eine Anweisung zu bearbeiten, können Sie sie zunächst auf eine der folgenden Weisen auswählen.

- Drücken Sie [*Strg*]+[*Umschalt*] oder [*Strg*]+[*Shift Lock*].
- Wählen Sie ein Menüelement im Testmenü-Ausschnitt.
- Klicken Sie eine Zeile im Script-Ausschnitt an, und drücken Sie [*F6*].
- Bewegen Sie sich im Script-Ausschnitt mit den Pfeiltasten, und drücken Sie [*F6*].

Sie können sich mit Hilfe der Maus im Dialogfeldausschnitt bewegen, d.h. den Mauszeiger positionieren und Ihre Auswahl treffen. Dies zeigt erst dann Wirkung, wenn Sie einen der folgenden Schritte ausführen:

- Drücken Sie auf die [*Eingabe*]-Taste.
- Mit [*Einfg*] oder *Menü/Neuer Menüeintrag* können Sie Änderungen eingeben und ein neues Element einfügen.

- Mit [*Strg*]+[*oben*] bzw. [*Strg*]+[*unten*] oder mit der Maus können Sie sich zu einer weiteren Anweisung bewegen.
- Mit [*Strg*]+[*P*] oder *Menü/Neues Pop-Up-Menü* können Sie ein neues Popup einfügen.
- Mit [*Strg*]+[*S*] oder *Menü/Neuer Separator* können Sie einen neuen Separator einfügen.

Die Optionen des Dialogfeldausschnitts sind in der Tabelle 16.2. angegeben.

Menüs bearbeiten

Sobald ein Menü im Menüeditor geladen ist, können Sie es in gewünschter Weise bearbeiten. Sie können neue Menübefehle, Popup-Menüs oder Separatoren zufügen oder einen beliebigen Teil des Menüs kopieren, löschen oder verschieben.

Neue Menübefehle einfügen

Um einen neuen Menübefehl in ein Menü (Popup-Menü, Menüelement oder Separator) einzufügen, müssen Sie den Mauszeiger im Script-Ausschnitt zur Zeile *über* derjenigen Zeile bewegen, wo Sie den Befehl einfügen wollen. Um den Befehl vor allen anderen einzufügen, markieren Sie die erste Zeile (MENU_1 oder den Namen der Menüressource).

Weitere Informationen zum Einfügen von Menübefehlen finden Sie auf den folgenden Seiten.

Nachdem Sie sich zu einer bestimmten Zeile für das neue Kommando entschieden und die entsprechende Zeile markiert haben, kann ein neuer Befehl durch Auswahl einer der Anweisungen aus Menü hinzugefügt werden.

Anweisung	Aktion
Neues Popup-Menü	Fügt ein neues Popup-Menü mit einem Eintrag ein
Neuer Menüeintrag	Fügt einen Eintrag ein
Neuer Separator	Fügt einen Separator ein
Neues Datei-Menü	Fügt eine vollständige generische Menüdatei hinzu
Neues Bearbeiten-Menü	Fügt ein Bearbeiten-Menü mit Kommandos zum Auschneiden, Kopieren und Einfügen ein
Neues Hilfe-Menü	Ein vordefiniertes Hilfe-Menü wird eingefügt
Auf doppelte IDs prüfen	Dies wird im folgenden beschrieben.

Wenn Sie einen Popup-Befehl einfügen, der zuerst auf der Menüleiste erscheinen soll (diese Position wird meist vom Datei-Menü eingenommen), markieren Sie die erste Zeile im Scriptfenster (MENU_1 oder der Name der Menütessource).

Menüelemente und Separatoren einfügen

Um ein neues Menüelement oder einen Seperaator ins Menü einzufügen,

1. Bestimmen Sie, wo die neue Anweisung im Menü erscheinen soll. Markieren Sie die Zeile über der gewünschten Position im Script-Ausschnitt. Der Menüeditor fügt den neuen Menüeintrag oder Separator unterhalb der markierten Zeile ein.
2. Drücken Sie [*Einfg*], oder wählen Sie *Menü/Neuer Menüeintrag*.
3. Geben Sie den Namen für das neue Menüelement ein.
4. Drücken Sie die [*Eingabe*]-Taste.

Drei Standardmenüs können Sie Ihrem Projekt ebenfalls hinzufügen:

- *Menü/Neues Dateimenü* fügt ein Dateimenü als erste Menü-Anweisung ein. Das Dateimenü enthält die Kommandos Neu, Öffnen, Speichern, Speichern unter, Drucken, Seitenlayout, Druckerinstallation und Beenden.
- *Menü/Neues Bearbeiten-Menü* fügt ein Bearbeiten-Menü als zweite Anweisung von links auf der Menüleiste ein. Das Bearbeiten-Menü besteht aus den Anweisungen Rückgängig, Ausschneiden, Kopieren und Einfügen.
- *Menü/Neues Hilfemenü* fügt ein Hilfemenü als letzte Menüanweisung ein (auf dem äußerst rechten Ende der Menüleiste). Das neue Hilfemenü enthält die Anweisungen Index, Tastatur, Befehle, Prozeduren , Hilfe verwenden und Info über.

Menüelemente bearbeiten

Ein neu erzeugtes Menüelement hat die generische Bezeichnung »Item«. Damit es verwendet werden kann, müssen Sie es bearbeiten.

Wenn Sie zum ersten Mal einen neuen Menüeintrag hinzufügen, wird er automatisch markiert, und Sie können ihn sofort bearbeiten. Wenn Sie aber stattdessen andere Menüelemente hinzufügen, müssen Sie zuerst ein Menüelement markieren, bevor Sie es bearbeiten können.

Benutzen des Dialogfeldausschnitts

Benutzen Sie die Maus, [*Tab*] oder [*Umschalt*]+[*Tab*], um auf das Feld zu gelangen, welches Sie bearbeiten wollen.

Die folgende Tabelle beschreibt die Auswahlmöglichkeiten, die Sie im Dialogfeldausschnitt haben.

Tabelle 16.2.: Optionen des Menüeditors im Dialogfeldausschnitt

Auswahl	**Beschreibung**
Text des Eintrags	Der Name des Popup-Menüs oder ein Menüeintrag (Kommando) und ggf. die Beschreibung des zugehörigen Tastenkürzels.
Hilfe	Ein Text, der den Menüeintrag beschreibt. Dieser Text wird in der Stringtabelle Ihres Projekts gespeichert. (Wenn Sie bislang keine Stringtabelle hatten, wird diese bei Benutzung der Option hinzugefügt.) Sie können diesen Text als Menühilfe auf einem Statusbalken verwenden.
ID-Wert	Eine einmalige ID für den Menüeintrag. Dieses Textfeld ist nicht verfügbar, wenn Sie ein Popup-Menü oder einen Separator ausgewählt haben.
Typ des Eintrags	Kann Popup-Menü, Menüeintrag oder Menü-Separator (Linie) sein.
Umbruch	Steuert das Format der Menübefehle in Menüleiste und in Popup-Menüs. Wählen Sie eine der Optionen:
- Kein Umbruch	Kein Unbruch vor dieser Anweisung.
- Menüleiste	Erzeugt eine neue Zeile in der Menüleiste. Im Popup-Menü wird eine neue Spalte erzeugt und eine senkrechte Linie zum Trennen der Spalten gezogen.
- Menü	Erzeugt eine neue Zeile in der Menüleiste oder eine neue Spalte im Popup-Menü.
- Hilfe	Bewegt die Popup Anweisung oder den Menüeintrag zum äußersten rechten Rand der Menüleiste. Benutzen Sie diese Option nur mit Anweisungen der obersten Stufe, die in der Menüleiste erscheinen.
Initialisierungsstatus	Steuert den Ursprungszustand der Menü-Anweisung. Wählen Sie eine der folgenden Optionen:
- Verfügbar	Im Ursprungszustand ist die Anweisung verfügbar. Benutzen Sie die Funktion *EnableMenuItem*, um den Zustand des Menüeintrags zu verändern.
- Nicht verfügbar	Im Ursprungszustand ist die Anweisung nicht verfügbar. Der Benutzer kann verfügbare und nicht verfügbare Anweisungen nicht unterscheiden, da sie im Menü gleich aussehen. Benutzen Sie die Funktion *EnableMenuItem*, um den Zustand des Menüeintrags zu verändern.
- Grau darstellen	Im Ursprungszustand ist die Anweisung nicht verfügbar, und der Anweisungstext wird grau dargestellt. Die graue Farbe zeigt dem Benutzer, daß die Anweisung im Augenblick nicht verfügbar ist. Benutzen Sie die Funktion *EnableMenuItem*, um den Zustand des Menüeintrags zu verändern.
- Markiert	Neben der Anweisung wird ein Häkchen als Markierungszeichen angezeigt. Diese Option wird gewählt, wenn die Anweisung wie ein Wechselschalter funktionieren und ursprünglich mit dem Häkchen markiert sein soll. Mit der *CheckMenuItem*-Funktion können Sie den Zustand des Menüeintrags verändern.
Taste	Steuert die Abkürzungen auf der Tastatur für den Menüeintrag. Sie können entweder einen Tastatur-Bezeichner (wie in WINDOWS.H definiert) eingeben, oder Sie benutzen *Menü/Wert für Tastenkürzel* (verfügbar im Schnellmenü, welches mit Rechtsklicken des Menüeintrags erreicht wird) und drücken die gewünschten Tasten. Um z.B. die Taste [*Pos1*] einem Menüeintrag zuzuweisen, markieren Sie den Menüeintrag, und rechtsklicken Sie darauf. Markieren Sie Tastenkürzel-Wert, und drücken Sie die Taste [*Pos1*]. VK_HOME erscheint im Tasten-Fenster, und der Invert menu item Modifier ist markiert. Drücken Sie [*ESC*], oder klicken Sie die Maus, um zum Bearbeiten zurückzukehren.

Tastenkürzel	Definieren der Tastenkürzel für einen Menüeintrag. Wählen Sie eine der folgenden Optionen:
- Alt	Benutzt [*Alt*] für das Tastenkürzel (z.B. [*Alt*]+[*W*])
- Umschalt	Benutzt [*Umschalt*] für das Tastenkürzel (z.B. [*Umschalt*]+[*F1*])
- Strg	Benutzt [*Strg*] für das Tastenkürzel (z.B. [*Strg*]+[*W*])
- Invert	Schaltet Flash feature aus. Dies ist eine eingebaute Windows-Funktion, die die dazugehörige Anweisung kurz hervorhebt, wenn der Benutzer ein Tastenkürzel drückt. Das zeigt dem Benutzer, auf welches Menü sich das Tastenkürzel bezieht. Invert menu item ist in der Voreinstellung bei Erzeugung eines Tastenkürzels eingeschaltet.
Tastentyp	Beschreibt den Typ des Tastenkürzels für einen Menüeintrag. Wählen Sie eine der Optionen:
- ASCII	Definiert das Tastenkürzel in Standard-ASCII-Tasten. Alle ASCII-Tasten müssen in Anführungsstrichen angegeben werden. Ein Karet ^ zeigt an, daß die Taste mit [*Strg*] verbunden ist. Normalerweise werden Tasten nicht einzeln als Tastenkürzel verwendet, sondern mit [*Alt*] oder [*Strg*] kombiniert, z.B. [*Alt*]+[*R*] oder [*Strg*]+[*L*].
- Virtuelle Taste	Eine virtuelle Taste ist normalerweise eine Funktionstaste, eine Pfeiltaste oder Bearbeitungstaste wie [*Pos1*] oder [*Ende*]. Windows hat vordefinierte Zeichen für virtuelle Tasten. Die Bezeichner beginnen mit VK_ und sind in WINDOWS.H definiert. Sie müssen die Bezeichner für virtuelle Tasten nicht nachschauen, wenn Sie Menu/Accelerator key value benutzen, um das Tastenkürzel einzufügen. Bei virtuellen Tasten ist es nicht vorgesehen, sie mit [*Strg*], [*Umschalt*] oder [*Alt*] zu kombinieren. Wenn Sie diese Tasten benutzen, müssen Sie die dazugehörigen Modifier-Fenster im Tastenkürzel Bearbeiten Dialogfenster markieren.

Menüeintragstext erstellen

Der Text des Eintrags ist der Menüname oder die Anweisung, die im Balken oder Menü erscheint. Wenn Text des Eintrags hervorgehoben ist, können Sie eine Zeichenkette direkt in das Fenster eingeben. [*Pos1*], [*Ende*] und die Pfeiltasten können zur Bewegung im Textfenster benutzt werden.

Wenn der Benutzer in der Lage sein soll, ein Menü oder eine Anweisung durch die Eingabe eines der Buchstaben im Text direkt aufzurufen, fügen Sie ein und (&) direkt vor dem Buchstaben ein. Windows zeigt bei der Textdarstellung diesen Buchstaben dann unterstrichen an.

Mehr Informationen über Tastenkürzel finden Sie in Kapitel 17.

Um [*Strg*]+[*F4*] mit der neuen Stored Order-Anweisung zu verbinden, fügen Sie Ihren Menüs das Tastenkürzel Text bei.

- Benutzen Sie das tab-Zeichen (\t), um den Menütitel vom Tastenkürzeltext zu trennen (z.B. &Stored Order\tStrg+F4).
- Benutzen Sie das Zeichen (\a) zur Ausrichtung nach rechts, um das Tastenkürzel an der rechten Seite auszurichten (z.B. &Stored Order\aStrg+F4).

Im allgemeinen wird in Windows-Anwendungen das Pluszeichen verwendet, um Tastenkombinationen darzustellen, z.B. [*Umschalt*]+[*Entf*] oder [*Strg*]+[*Umschalt*]+[*F4*].

Eingabe von Eintrags-IDs

Alle Menüeinträge müssen eindeutig identifizierbar sein. Bei Erzeugen eines neuen Eintrags wird diesem automatisch eine ID zugewiesen, die von allen anderen Einträgen in der Menüressource verschieden ist. Sie können den Wert übernehmen, oder Sie können ihn durch eine einmalige Zahl oder Namensgebung ersetzen.

Mehr Informationen über Bezeichner finden Sie weiter vorn.

Wenn Sie einen Namen eingeben, wird zunächst geprüft, ob bereits ein Bezeichner dieses Namens existiert. Ist dies nicht der Fall, werden Sie in einem Dialogfenster gefragt, ob Sie einen Bezeichner erzeugen wollen. Klicken Sie auf OK. Wenn Sie den Namen als ID eines Eintrags benutzen wollen, müssen Sie einen Bezeichner erzeugen. Ab dieser Stelle müssen Sie einen Bezeichner erzeugen, wie es bereits in Kapitel 14 beschrieben wurde.

Menü-Anweisungen verschieben und kopieren

Sie können die Befehle Ausschneiden, Kopieren und Einfügen des Menüs Bearbeiten verwenden, um Ihre Anweisungen im Scriptfenster des Editors zu verschieben und zu kopieren.

Um eine Anweisung zu verschieben, markieren Sie sie und wählen *Bearbeiten/Ausschneiden*. Sie können nicht die letzte (einzige) Anweisung im Scriptfenster ausschneiden, da immer mindestens ein Menüeintrag, Popup-Menü oder Separator im Scriptfenster vorhanden sein muß. Um Ihre ausgeschnittene oder kopierte Anweisung einzutragen, markieren Sie die Zeile über der Zeile, in die Sie eintragen wollen, und wählen *Bearbeiten/Einfügen*.

Um eine Anweisung zu kopieren, markieren Sie sie und wählen *Bearbeiten/Kopieren*. Die markierte Anweisung bleibt an ihrer Position im Scriptfenster.

Menü-Anweisungen löschen

Markieren Sie die zu löschende Anweisung. Drücken Sie [*Entf*], oder wählen Sie *Bearbeiten/Löschen* oder *Bearbeiten/Ausschneiden*.

Beachten Sie folgendes, wenn Sie Anweisungen löschen:

- Löschen Sie eine Popup-Anweisung, wird auch der Popup-Befehl sowie alle Einträge in diesem Menü gelöscht.
- _END Popup_ Anweisungen können nicht gelöscht werden.
- Sie können eine Anweisung nicht löschen, wenn sie die einzige im Scriptfenster ist, da immer mindestens ein Menüeintrag, Popup-Menü oder Separator im Scriptfenster vorhanden sein muß.

Ad hoc-Menüs erzeugen

Ein Ad hoc-Menü kann an beliebiger Stelle des Anwendungsfensters erscheinen und ist nicht an die Menüleiste gebunden.

Ein Ad hoc-Menü muß als eigene Ressource in der Projektdatei gespeichert werden.

Um ein Ad hoc-Menü zu erzeugen,

1. Können Sie mit *Ressource/Neu* ins Neue Ressource-Dialogfenster gelangen. Wählen Sie dort Menü.
2. Mit *Ansehen/Ansehen* als Popup können Sie sich das Ad hoc-Menü ansehen, wie es während der Laufzeit dargestellt werden wird.
 Wenn Sie sich das Menü im Testmenü-Auschnitt ansehen, wird es wie an die Menüleiste gebunden aussehen. Benutzt Ihr Programm jedoch die *TrackPopupMenu*-Funktion in korrekter Weise, wird das Menü während der Laufzeit an beliebiger Stelle darstellbar sein.
3. Markieren Sie die erste Zeile im Scriptfenster (MENU_1 oder den Namen der Menüressource).
4. Drücken Sie [*Einfg*], um mindestens einen Menüeintrag zu Beginn des Scriptfensters einzufügen.
5. Markieren Sie die Zeile POPUP »Popup« im Scriptfenster.
6. Mit [*Entf*] können Sie diese Menüanweisung, die zugehörigen Menüeinträge und die Anweisung _End Popup_ löschen.
7. Fügen Sie weitere Menüeinträge ein, sofern Sie das wünschen.
8. Bearbeiten Sie die Menüeinträge im Dialogfenster.
9. Speichern Sie Ihr Projekt.

Menüs testen

Während des Arbeitens mit dem Menüeditor können Sie Ihr Menü sofort testen. Das Testmenü wird immer auf den neuesten Stand gebracht, während Sie Änderungen vornehmen, und so können Sie sich das Menü jederzeit anzeigen lassen, um dessen augenblickliche Erscheinungsform zu prüfen. Wählen Sie einen Menüeintrag, wird er hervorgehoben, und im Dialogfeldausschnitt erscheinen seine Eigenschaften. Sie können dies durch die Option *Menü/Testmenü verfolgen* ausschalten.

Der Menüeditor hat ein eingebautes Programm zur Fehlersuche, den Debugger. Er kann benutzt werden, um das Menü nach uneindeutigen, d.h. mehrfach zugewiesenen IDs von Menüeinträgen zu untersuchen. Mit *Menü/Auf doppelte IDs prüfen* sucht der Editor nach mehrfach zugewiese-

nen IDs und zeigt ein Dialogfenster mit der Nachricht »Doppelten ID-Wert gefunden«, wenn einer gefunden wird.

Schließen Sie dieses Fenster, und im Menüeditor erscheint die Anweisung hervorgehoben, auf die die mehrfach zugewiesene ID Bezug nimmt. Sie müssen einen der folgenden Schritte durchführen:

Mit *Ressource/Bearbeiten als Text* können Sie sich die IDs des Menüs ansehen.

- Ist die ID des Eintrags eine Zahl, geben Sie eine neue Zahl an, die bisher keiner ID zugewiesen ist.
- Ist die ID des Eintrags ein Bezeichner, enthält das Fenster Eintrag ID eine Zeichenkette und das Fenster rechts vom Fenster Eintrag ID eine Zahl. Um den Wert des Bezeichners zu ändern,

1. Lassen Sie sich mit *Ressource/Bezeichner* das Bezeichner-Dialogfenster anzeigen.
2. Durchsuchen Sie die Bezeichner-Liste, bis Sie den gewünschten gefunden haben.
3. Klicken Sie die [*Ändern*]-Taste, und geben Sie einen Wert ein, der keinem anderen ID zugewiesen ist.
4. Klicken Sie auf OK, oder drücken Sie die [*Eingabe*]-Taste, um den Wert zu ändern.
5. Klicken Sie in das Menüeditor-Fenster, um mit der Bearbeitung des Menüs fortzufahren. (Sie können das Bezeichner-Dialogfenster für spätere Anwendungen geöffnet lassen.)

Wenn Sie z.B. den Wert 101 den zwei Bezeichnern *wmnu_List* und *wmnu_Asc* zuweisen, wird mit *Menü/Auf doppelte IDs prüfen* die Nachricht »Duplicate command value found« ausgegeben und im Menüeditor *wmnu_Asc* hervorgehoben (da er als zweiter eingegeben wurde). Wie oben beschrieben würden Sie dann den Wert von *wmnu_Asc* neu bestimmen und einen anderen Wert wählen als 101, 102, 103 oder 104, da dies die Werte der anderen Bezeichner im Menü sind.

Bearbeiten von Menüressourcen-Scripts

Zur Bearbeitung des Ressourcen-Scripts eines Menüs markieren Sie den Menünamen im Projektfenster, indem Sie darauf klicken, und mit *Ressource/Bearbeiten* als Text wird das Ressourcen-Script im internen Texteditor angezeigt.

Die Beschreibung des internen Texteditors finden Sie weiter vorn.

Um zum Beispiel das Ressourcen-Script für das Beispielmenü zu bearbeiten, welches im nächsten Abschnitt erzeugt werden wird, können Sie das Projekt öffnen, welches das Menü enthält, das Menü markieren. Mit *Ressource/Bearbeiten* als Text wird der interene Texteditor aufgerufen und zeigt den Quellcode wie folgt an:

Vgl. Sie die Beschreibung der Ressourcen-Script-Optionen für Menüs im online Hilfsindex.

```
MENU_1 MENU
BEGIN
    POPUP "&Widgets"
    BEGIN
        MENUITEM "&List\tCtrl+L", wmnu_List
         MI "&Add...\tCtrl+A", wmnu_Add
         MI SEPARATOR
         PO "A&rrange List"
         BEGIN
             MI "&Ascending\tCtrl+F2", wmnu_Asc
             MI "&Descending\tCtrl+F3", wmnu_Dasc
        END
    END
END
```

Mit dem Editor können Sie Änderungen direkt im Ressourcen-Script vornehmen. Um zum Beispiel zwei der Speicheroptionen von LOADONCALL und MOVEABLE (die Standardeinstellungen) zu PRELOAD und FIXED zu ändern,

Mit Resource/Memory Options können Sie die Änderungen ebenfalls durchführen.

1. Ändern Sie mit dem Texteditor die erste Zeile des Scripts wie folgt:

   ```
   MENU_1 MENU PRELOAD FIXED
   ```

2. Mit Compilieren/Compilieren können Sie Ihre Eingabe compilieren und somit überprüfen lassen.

 Das Menü Compilieren ist nur verfügbar, wenn Sie sich im Texteditor befinden. Wenn Sie zum Menüeditor zurückkehren und mit Ressource/Bearbeiten als Text wieder zum Texteditor umschalten, werden Sie sehen, daß eine der Standard-Speichereinstellungen ins Script eingefügt wurde. Diese erste Zeile des Scripts sieht nun wie folgt aus:

   ```
   MENU_1 MENU PRELOAD FIXED DIISCARDABLE
   ```

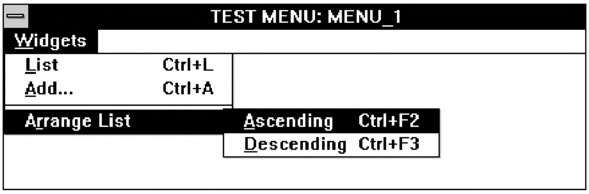
3. Wollen Sie den Menüeditor verlassen, wählen Sie die Schließen-Anweisung des Control menu-Fensters des Texteditorfensters. Es wird nachgefragt, ob compiliert werden soll. Klicken Sie auf Ja, und das Menü wird compiliert. Danach befinden Sie sich im Projektfenster.

➥ Verschwenden Sie keine Zeit, indem Sie Ihr Ressourcen-Script formatieren oder mit Kommentaren versehen. Während der Compilierung wird der Text umformatiert und alle Kommentare gelöscht.

Ein Beispielmenü

Dieser Abschnitt beschreibt die Erzeugung eines einfachen Popup-Menüs namens Widgets, zuerst mit dem Menüeditor und dann mit dem Texteditor.

Abbildung 16.1: Beispielmenü



Die ersten beiden Anweisungen im Menü (List und Add) erlauben dem Benutzer, bestehende Fenster auflisten zu lassen oder neue hinzuzufügen. Die dritte Anweisung, Arrange List, erzeugt ein Popup-Menü mit zwei zusätzlichen Kommandos (Ascending and Descending), mit denen der Benutzer die Art der Sortierung in der Liste festlegen kann.

Widgets und Arrange List sind Popup-Befehle. Widgets zeigt das Widget-Menü an. Arrange List erzeugt das Kaskaden-Menü mit den Ascending und Descending Befehlen.

List, Add, Ascending und Descending sind Menüeinträge (oder Befehle). Klickt der Benutzer eines der Kommandos an, führt die Anwendung die zugehörige Aktion aus. Die Ellipsen (drei Punkte) nach dem Add-Befehl zeigen an, daß die Anwendung ein Dialogfenster öffnet, wenn der Benutzer diesen Eintrag wählt.

Beachten Sie, daß mit dem Separator die List- und Add-Befehle zu einer Gruppe getrennt vom Arrange List - Popup-Befehl zusammengefaßt sind.

Das Beispielmenü erzeugen

Mit dem Menüeditor haben Sie die Möglichkeit, das Beispielmenü viel einfacher als mit dem Texteditor zu erzeugen. Unter anderem verwaltet der Menüeditor die IDs des Menüs für Sie und speichert die Bezeichner in einer getrennten Bezeichner-Datei (sofern Sie eine erzeugt haben). Zusätzlich können Sie das Menü schon während der Erzeugung testen.

Um das Beispielmenü zu erzeugen,

Kapitel 14 beschreibt, wie Sie ein Projekt öffnen.

1. Stellen Sie sicher, daß Sie ein Projekt geöffnet haben.
2. Bei geöffnetem pp wählen Sie Ressource/Neu.
3. Das Neue Ressource-Dialogfenster wird angezeigt. Durchsuchen Sie die Ressourcentypen-Liste, bis Sie MENU sehen, und doppelklicken Sie darauf.

Eine neue Menüressource wird dem Projektfenster hinzugefügt, und das neue Menü wird im Menüeditor mit der ersten Zeile im Scriptfenster hervorgehoben dagestellt.

4. Um den Befehlsnamen der Menüanweisung von Popup zu Widgets zu ändern, geben Sie &Widgets in das Eingabefeld Eintrags Text im Dialogfenster ein und drücken die [*Eingabe*]-Taste.

Der Menüeditor bringt sowohl das Script als auch das Testmenü auf den neuesten Stand.

Im Testmenü ist das *W* in Widgets unterstrichen, um Ihnen anzuzeigen, daß mit dem Tastenkürzel [*Alt*]+[*W*] dieses Menü aufgerufen werden kann.

Befehle in das Menü einfügen

Als nächstes fügen Sie die Befehle in das Widget-Menü ein.

1. Um den ersten Menüeintrag umzubenennen und um Text einzutragen, der das Tastenkürzel beschreibt, drücken Sie [*Strg*]+[*unten*], um die zweite Zeile im Scriptfenster (MENUITEM "Item") hervorzuheben. Geben Sie &List\tCtrl+L in das Eingabefeld Eintragstext ein.

2. Mit TAB gehen Sie zum Eingabefeld ID des Eintrags und geben *wmnu_List* ein, um einen Bezeichner für diesen Befehl anzugeben. Drücken Sie die [*Eingabe*]-Taste.

 Sie werden gefragt, ob Sie einen neuen Bezeichner erzeugen wollen. Drücken Sie die [*Eingabe*]-Taste, und das New Identifier-Dialogfenster erscheint. Drücken Sie die [*Eingabe*]-Taste erneut, um den angezeigten Wert zu bestätigen.

3. Bei hervorgehobenem Listkommando fügen Sie einen neuen Menübefehl entweder durch Drücken auf [*Einfg*] oder durch Menu/New Item ein.

4. Geben Sie &Add...\tCtrl+A in das Menüeintrag-Eingabefeld, um den Text für den neuen Menüeintrag zu ändern.

5. Erzeugen Sie einen Bezeichner für den Eintrag, indem Sie mit [*TAB*] zum Feld Item ID gehen und *wmnu_Add* eingeben. Drücken Sie die [*Eingabe*]-Taste, und reagieren Sie wie zuvor auf die Eingabeaufforderungen.

6. Mit [*Strg*]+[*S*] wird der Separator nach dem Add-Befehl eingefügt.

7. Bei markiertem Add Befehl drücken Sie [*Strg*]+[*P*], um ein neues Popup-Menü anzufügen. Den Text ändern Sie in A&rrange List.

Da Sie zusätzliche Befehle anzeigen lassen wollen, wenn der Benutzer Arrange List wählt, definieren Sie es als Popup-Befehl und nicht als einen weiteren Menüeintrag. Ein Popup-Kommando inmitten einen Menüs erzeugt ein Kaskaden-Menü.

Einfügen von Anweisungen in das Arrange List-Menü

Um die zwei Anweisungen im Arrange List - Popup-Menü zu erzeugen,

1. Drücken Sie [*Strg*]+[*unten*], bis der erste Menüeintrag im Arrange List-Menü hervorgehoben ist.
2. Ändern Sie den Menüeintrag von »Item« zu »&Ascending\tCtrl+F2«.
3. Erzeugen Sie den Bezeichner *wmnu_Asc* für den Befehl.
4. Drücken Sie [*Einfg*], um einen neuen Menüeintrag hinter »&Ascending« anzufügen, und ändern Sie ihn zu »&Descending\tCtrl+F3«.
5. Erzeugen Sie den Bezeichner *wmnu_Desc* für den Befehl.
6. Speichern Sie Ihr Projekt.

Das Menü testen

Testen Sie das Menü, indem Sie auf das Widgets-Kommando im Testmenü klicken und das Arrange List-Kommando herunterziehen. Ihr Menü sollte wie in Abbildung 16.1 aussehen.

Sie können nach mehrfacher Verwendung von Werten für IDs im Menü durch Menu/Check Duplicates suchen lassen.

- Gibt es keine Mehrfachverwendung, wird »Keine Doppelbezeichnungen gefunden« angezeigt.
- Wird eine Mehrfachverwendung festgestellt, erscheint »Doppelter Kommandoname gefunden«. Im Abschnitt »Menüs testen« in diesem Kapitel erfahren Sie, wie das korrigiert werden kann.

K A P I T E L 17

Tastenkürzel definieren

Ein *Tastenkürzel*, auch Hot Key genannt, ist eine Tastenkombination, die der Anwender drückt, um eine bestimmte Aufgabe ausführen zu lassen. Ein Tastenkürzel ist ein Ersatz für einen Menübefehl. Wie ein Menübefehl erzeugt es eine WM_COMMAND- oder WM_SYSCOMMAND-Botschaft, die der Anwendung den nächsten notwendigen Schritt mitteilt.

Gewöhnlich werden Tastenkürzel als Alternative zu Menübefehlen erzeugt. Wenn Sie z.B. das Menü Bearbeiten in vielen Windows-Anwendungen öffnen, sehen Sie oftmals die Tastenkürzel: [*Alt*]+[*Rück*] (Rückgängig), [*Umschalt*]+[*Entf*] (Ausschneiden), [*Strg*]+[*Einfg*] (Kopieren) und [*Umschalt*]+[*Einfg*] (Einfügen).

Sie speichern die Tastenkürzeldefinitionen in einer Tastenkürzeltabelle bzw. Tastenkürzel-Ressource. Jeder Eintrag in der Tabelle beschreibt das Tastenkürzel als Tastenkombination (die der Anwender drückt) und das Befehl, der dadurch ausgelöst wird. Sie können mehrere Tastenkürzel-Ressourcen für verschieden Teile Ihres Menüs definieren.

Der Tastenkürzeleditor kann Tastenkürzel für Ihre Anwendung erzeugen und bearbeiten. Die Arbeit mit Tastenkürzeln umfaßt fünf grundlegende Schritte:

1. Starten Sie den Tastenkürzeleditor.
2. Starten Sie den Menüeditor. Im Menüeditor können Sie Tastenkürzel für Menüeinträge definieren.
3. Erzeugen oder bearbeiten Sie die Tastenkürzeltabelle.
4. Testen Sie die Tastenkürzeltabelle auf doppelte Tasten.
5. Speichern Sie die Tastenkürzeltabelle.

Tastenkombinationen in der Tastenkürzeltabelle

Sie können als Tastenkombinationen in Ihrer Tastenkürzeltabelle *ASCII-Tasten* und *virtuelle Tasten* verwenden.

- Eine ASCII-Taste kann auf dem Bildschirm dargestellt werden – typischerweise ein alphanumerisches Zeichen oder ein Satzzeichen.
- Eine virtuelle Taste ist eine Funktionstaste, eine Pfeiltaste oder eine Taste zum Bearbeiten wie [*Pos1*] oder [*BildOben*]. Obwohl diese Tasten manchmal durch Zeichen auf dem Bildschirm dargestellt werden, gibt es keinen Standard, der festlegt, welches Zeichen erscheint.

ASCII-Tasten

Alle ASCII-Tasten sind von Anführungsstrichen umgeben. Ein Cursor (^) zeigt an, daß die Taste mit der [*Strg*]-Taste zusammen gedrückt werden muß. Im Attributfeld zeigt das Auswahlfeld *Alt* an, ob die Taste mit der [*Alt*]-Taste kombiniert ist.

So würden z.B. [*Strg*]+[*W*] und [*Strg*]+[*Alt*]+[*W*] im Script-Ausschnitt als ^W angezeigt, und der Unterschied wäre nur im Dialogfeld zu sehen, da [*Strg*]+[*Alt*]+[*W*] unter Alt markiert wäre.

➥ Normalerweise werden keine einzelnen ASCII-Zeichen als Tastenkürzel verwendet – stattdessen werden sie mit [*Alt*] oder [*Strg*] kombiniert ([*Strg*]+[*L*] oder [*Alt*]+[*L*] statt [*L*]).

Virtuelle Tasten

Windows benutzt vordefinierte Bezeichner für virtuelle Tasten, so z.B. VK_BACK für [*Rück*] und VK_F1 für [*F1*]. Diese Bezeichner, die alle mit VK_ anfangen, sind in WINDOWS.H definiert.

Die Auswahlfelder für Umschalttasten im Attributfeld zeigen, ob die Taste mit [*Strg*], [*Alt*], [*Umschalt*] oder mit irgendeiner Kombination dieser drei Tasten gedrückt werden muß.

Sie könnten z.B. zwei VK_F1-Tastenkürzel in Ihrer Tabelle haben. Das erste könnte [*Strg*]+[*F1*] sein. (Dann wäre das Feld *Strg* markiert.) Das andere könnte [*Umschalt*]+[*F1*] sein. (Das Feld *Umschalt* wäre markiert.)

➥ Sie müssen keinen der Bezeichner für virtuelle Tasten nachsehen, wenn Sie den Tastenkürzeleditor im Tastenwert-Modus betreiben und direkt die Tasten eingeben, da dieser die richtigen Werte automatisch einfügt.

Den Tastenkürzeleditor aufrufen

Wie der Tastenkürzeleditor aufgerufen wird, hängt davon ab, ob Sie eine neue Tastenkürzeltabelle erzeugen oder eine bereits vorhandene bearbeiten wollen.

Neue Tastenkürzeltabelle erzeugen

Kapitel 14 beschreibt, wie ein Projekt geöffnet wird.

Sie können eine neue Tastenkürzeltabelle in einem neuen oder bereits bestehenden Projekt erzeugen. Um den Tastenkürzeleditor aufzurufen und eine neue Tastenkürzeltabelle zu erzeugen, gehen Sie wie folgt vor:

1. Mit *Datei/Neues Projekt* erzeugen Sie ein neues oder mit *Datei/Projekt öffnen* rufen Sie ein bestehendes Projekt auf.
2. Mit *Ressource/Neu* wird das Dialogfenster *Neue Ressource* angezeigt.
3. Doppelklicken Sie im Dialogfenster auf ACCELERATOR im Listenfeld Ressourcen-Typ.

Der Tastenkürzeleditor erscheint mit einer Tastenkürzeltabellenvorlage, die Sie bearbeiten können.

Bestehende Tastenkürzeltabelle bearbeiten

Öffnen Sie das Projekt, das die Tastenkürzel-Ressource enthält, die Sie bearbeiten wollen. und laden Sie die betreffende Tastenkürzeltabelle mit einem der beiden Schritte in den Tastenkürzeleditor:

- Doppelklicken Sie im Projektfenster auf den Namen der Tastenkürzel-Ressource.
- Markieren Sie den Namen der Tastenkürzel-Ressource im Projektfenster, und wählen Sie *Ressource/Bearbeiten*.

Der Resource Workshop öffnet den Tastenkürzeleditor mit der von Ihnen gewählten Tastenkürzel-Ressource.

Arbeiten mit aktivem Menüeditor

Bei der Bearbeitung von Tastenkürzeln ist es gewöhnlich sinnvoll, parallel den Menüeditor aufzurufen und das zur Tastenkürzeltabelle gehörende Menü zu laden. Auf diese Weise können Sie die für die Definition der Tastenkürzel benötigten Einträge für Befehlstext und ID-Wert sehen. Es wird später in diesem Kapitel erklärt, daß jedem Tastenkürzel ein Bezeichner zugeordnet sein muß, der einem Menübefehl entspricht. Eine Beschreibung, wie Tastenkürzel Menübefehlen zugeordnet werden, finden Sie in Kapitel 16.

Arbeiten mit dem Tastenkürzeleditor

Die Arbeitsfläche des Tastenkürzeleditor ist in zwei Ausschnitte unterteilt, den Dialog- und den Script-Ausschnitt. Sie können mit der Maus oder [*F6*] zwischen den beiden Ausschnitten wechseln.

Der Script-Ausschnitt

Der Script-Ausschnitt zeigt Ihnen in script-ähnlicher Form alle Tastenkürzel der Tabelle an. Die oberste Zeile des Ausschnitts zeigt den Namen der Tastenkürzeltabelle. Darunter befinden sich die Einträge für die Tastenkürzel, welche aus zwei Teilen bestehen:

Am Anfang dieses Kapitels finden Sie eine Erklärung des Unterschieds zwischen ASCII-Tasten und virtuellen Tasten.

- Der erste Teil beschreibt die Taste, die als Tastenkürzel genutzt wird. Es handelt sich um eine ASCII-Taste oder virtuelle Taste.
- Der zweite Teil ist der ID-Wert des Befehls, der mit dem Tastenkürzel erzeugt wird. Der Eintrag ist eine Integerzahl oder der Name eines Bezeichners.

Mit der Maus oder den Pfeiltasten können Sie ein Tastenkürzel markieren. Mit Klicken auf die rechte Maustaste oder mit [*Alt*]+[*F10*] wird im Script-Ausschnitt ein lokales Menü angezeigt.

Der Dialogfenster-Ausschnitt

Wenn Sie im Script-Ausschnitt ein Tastenkürzel markiert haben, sehen Sie im Dialogfenster-Ausschnitt die Einstellungen zu dem Tastenkürzel. Im Dialogfenster-Ausschnitt können Sie nun Änderungen vornehmen, wie andere Tastenkombinationen für das Tastenkürzel definieren oder das Tastenkürzel einem anderen Befehl zuweisen.

Im Dialogfenster-Ausschnitt benutzen Sie die Maus oder [*Strg*]+[*Oben*] oder [*Strg*]+[*Unten*], um ein neues Tastenkürzel zu markieren.

Die Einstellungen in den Feldern können mit der Maus oder durch folgende Tasten verändert werden:

- Mit [*Tab*] (vorwärts) und mit [*Umschalt*]+[*Tab*] (rückwärts) aktivieren Sie die einzelnen Dialoggruppen des Dialogfenster-Ausschnitts. Jedes der Auswahlfelder ist ein Tabulatorstop.

 Wenn Sie mit [*Tab*] das Feld *Taste* aktivieren, schaltet der Tastenkürzeleditor in den Tastenwert-Modus. Zur Definition eines neuen Tastenkürzels müssen Sie in diesem Modus nur die gewünschten Tasten bzw. Tastenkombinationen drücken. Durch Klicken oder mit [*Alt*]+[*Esc*] beenden Sie den Tastenwert-Modus. Der Tastenwert-Modus wird nachfolgend näher beschrieben.

- Mit den Pfeiltasten erreichen Sie die Auswahlfelder in der Gruppe *Tastentyp*. Mit der Leertaste erreichen Sie die Auswahlfelder in der Gruppe *Umschalttasten*.

Ihre Einstellungen werden wirksam, wenn Sie die Eingabetaste drücken (um ein Tastenkürzel zu ändern) oder die [*Einfg*]-Taste (um ein neues Tastenkürzel einzufügen), oder wenn Sie ein neues Tastenkürzel im Script-Ausschnitt markieren.

Folgende Tabelle beschreibt die Auswahlmöglichkeiten im Dialogfenster-Ausschnitt:

Tabelle 17.1: Elemente des Dialogfenster-Ausschnitt im Tastenkürzeleditor

Auswahl	Beschreibung
Befehl	Der ID-Wert (Integer oder Bezeichner) des Befehls, der mit dem Tastenkürzel verbunden ist. Der Wert muß dem Wert entsprechen, der der Menü-Ressource zugeordnet ist.
Taste	Das Tastenkürzel. Tastenkürzel können manuell oder im Tastenwert-Modus eingegeben werden. Im manuellen Modus geben Sie die Taste entweder in Anführungszeichen ein, wenn es sich um eine ASCII-Taste handelt, oder ihren Bezeichner, wenn es sich um eine virtuelle Taste handelt. Im Tastenwert-Modus drücken Sie auf die gewünschte Taste. Der Tastenkürzeleditor entscheidet, ob es sich um eine ASCII- oder virtuelle Taste handelt, und trägt diese entsprechend ein. Wenn Sie mit [*Tab*] auf dieses Feld gehen, sind Sie automatisch im Tastenwert-Modus.
Tastentyp	ASCII oder virtuell. Im Tastenwert-Modus setzt der Tastenkürzeleditor die folgenden Felder von selbst.
– ASCII	Das Tastenkürzel benutzt eine ASCII-Taste.
– Virtuelle Taste	Das Tastenkürzel benutzt eine virtuelle Taste.
Umschalttasten	Ein Satz von Auswahlfeldern, deren Bedeutung im weiteren erläutert wird:
– Alt	Das Tastenkürzel enthält die [*Alt*]-Taste (zum Beispiel [*Alt*]+[*W*]).
– Umschalt	Das Tastenkürzel enthält die [*Umschalt*]-Taste (zum Beispiel [*Umschalt*]+[*F1*]).
– Strg	Das Tastenkürzel enthält die [*Strg*]-Taste (zum Beispiel [*Strg*]+[*F3*]).
– Menüeintrag invertieren	Bei Verwendung des Tastenkürzels wird der dazugehörige Eintrag im Menü hervorgehoben (invertiert).

Eine Tastenkürzeltabelle bearbeiten

Nachdem Sie eine Tastenkürzeltabelle in den Tastenkürzeleditor geladen haben, können Sie sie bearbeiten. Sie können Tastenkürzel definieren, verändern und neue Tastenkombinationen eingeben, indem Sie einfach die gewünschte Tastenkombination drücken. Zudem können Sie Tastenkürzel kopieren und löschen sowie die Tabelle auf doppelte Tastenbelegung prüfen.

Ein Tastenkürzel einfügen

Um ein neues Tastenkürzel in die Tabelle einzufügen, drücken Sie die [*Einfg*]-Taste oder wählen *Tastenkürzel/Neuer Eintrag*. Der neue Eintrag erscheint im Script-Ausschnitt unterhalb der zuletzt markierten Zeile mit

den Vorgaben 0 für Taste und einem einmaligen Integerwert für die Befehls-ID.

Nach dem Einfügen des Eintrags ist automatisch der Dialogfenster-Ausschnitt aktiv.

Ein Tastenkürzel markieren

Mit einer der folgenden Methoden können Sie Tastenkürzel markieren:

- Im Script-Ausschnitt wird mit [*Strg*]+[*Oben*] bzw. [*Strg*]+[*Unten*] ein Tastenkürzel markiert und zum Dialogfenster-Ausschnitt umgeschaltet.
- Klicken Sie einen Eintrag im Script-Ausschnitt an. Mit [*F6*] können Sie in den Dialogfenster-Ausschnitt gelangen.
- Wenn Sie im Script-Ausschnitt arbeiten, markieren Sie einen Eintrag mit den Pfeiltasten. Mit [*F6*] können Sie in den Dialogfenster-Ausschnitt wechseln.

Den Dialogfenster-Ausschnitt benutzen

Sie finden im Dialogfenster-Ausschnitt Textfelder, Schaltfelder und Auswahlfelder zur genauen Beschreibung von Tastenkürzeln. Die Auswahlmöglichkeiten im Dialogfenster-Ausschnitt sind in Tabelle 17.3 beschrieben.

Den Befehlswert eingeben

Im Textfeld *Befehl* geben Sie den ID-Wert des Befehls ein, der mit dem Tastenkürzel verknüpft ist. Der Wert kann eine Integerzahl oder ein Bezeichner sein.

Die Zeichenfolge des Befehls wird automatisch hervorgehoben und enthält den aktuellen ID-Wert, wenn Sie ein neues Tastenkürzel mit [*Einfg*] einfügen oder ein bestehendes mit [*Strg*]+[*Oben*] oder [*Strg*]+[*Unten*] markieren. Geben Sie den ID-Wert direkt in das Textfeld ein.

Wenn Sie das Tastenkürzel mit einem bereits definierten Menübefehl verbinden, öffnen Sie den Menüeditor und laden die entsprechende Menü-Ressource. Merken Sie sich den ID-Wert des Befehls, den Sie mit dem Tastenkürzel verbinden wollen, und tragen Sie diesen Wert in das Textfeld *Befehl* ein.

Beachten Sie folgendes, wenn Sie Bezeichner verwenden:

- Erscheint das Dialogfenster zur Eingabe eines neuen Bezeichners, nachdem Sie einen vorhandenen Bezeichner eingegeben haben, haben Sie sich wahrscheinlich vertippt. Klicken Sie auf *Nein,* und überprüfen Sie Ihre Eingabe.
- Wollen Sie erst später ein Menüelement zum Tastenkürzel entwerfen, so können Sie eine neuen Bezeichner eingeben. Wenn Sie gefragt werden, ob Sie einen neuen Bezeichner erzeugen wollen, klicken Sie auf *Ja* oder

drücken die Eingabetaste und geben dann einen einmaligen ID-Wert im Dialogfenster *Neuer Bezeichner* ein.

Wird Ihnen kein Dialogfenster zum Erstellen eines neuen Bezeichners angezeigt, besteht dieser Bezeichner bereits. Geben Sie dann einen noch nicht benutzten Bezeichner ein, bevor Sie mit der Arbeit fortfahren.

Tastenkürzel angeben

Im Textfeld *Taste* geben Sie die Taste bzw. Tastenkombination an, die als Tastenkürzel verwendet werden soll. Die Tastenkombination sollte den Tastenkürzeln anderer Windows-Anwendungen nicht widersprechen. Benutzen Sie daher keine Windows-Tastenkombinationen wie [*Strg*]+[*Esc*]. Richtlinien zur Auswahl von angemessenen Tastenkombinationen finden Sie im Handbuch *System Application Architecture Common User Access Advanced Interface Design Guide* der Firma IBM.

Eine Tastenkombination kann im manuellem oder Tastenwert-Modus eingegeben werden.

Tastenwert-Modus: Im Tastenwert-Modus wird jede Taste oder Tastenkombination, die Sie drücken, automatisch als Tastenkombination eines Tastenkürzels in das Textfeld Taste eingetragen. Der Tastenkürzeleditor entscheidet, ob als Taste eine ASCII-Taste oder eine virtuelle Taste verwendet wurde und markiert entsprechend die zugehörigen Schaltfelder unter *Tastentyp*. Ebenso automatisch werden ggf. die Auswahlfelder in der Gruppe *Umschalttasten* gesetzt.

Manueller Modus: Im manuellen Modus müssen Sie alle benötigten Informationen selbst eintragen. Sie entscheiden, ob die Taste eine ASCII-Taste oder virtuelle Taste war. Bei einer virtuellen Taste muß Ihnen der dazugehörige Windows-Bezeichner bekannt sein. Sie geben ggf. den Bezeichner in Großbuchstaben ein und markieren das Schaltfeld unter *Tastentyp* (ASCII-Taste oder virtuelle Taste). Entsprechend der Tastenkombination setzen Sie die Auswahlfelder Alt, Strg und Umschalt.

Menübefehl hervorheben

Windows verfügt über eine Funktion, die einen Befehl in der Menüleiste hervorhebt, sobald ein Tastenkürzel für einen Befehl gedrückt wurde, der mit diesem Menüpunkt verbunden ist.

In vielen Windows-Anwendungen wird, wenn Sie einen Block markiert und [*Umschalt*]+[*Entf*] gedrückt haben (entspricht *Bearbeiten/Ausschneiden*), der Menüpunkt *Bearbeiten* kurzzeitig hervorgehoben. Auf diese Weise weiß ein Anwender, wo dieser Befehl im Menü zu finden wäre.

Diese Funktion (Menüeintrag invertieren genannt) ist standardmäßig eingeschaltet, wenn Sie ein Tastenkürzel erzeugen. Sie können die Eigenschaften unterdrücken, indem Sie in der Dialoggruppe Umschalttasten das Auswahlfeld *Menüeintrag invertieren* ausschalten.

Prüfen auf doppelte Tastenbelegung

Um sicher zu sein, daß dieselbe Tastenkombination nicht mehrfach verwendet wurde, können Sie mit dem Resource Workshop eine Tastenkürzeltabelle wie folgt nach doppelten Tastenbelegungen durchsuchen:

1. Wählen Sie bei geöffneter Tastenkürzeltabelle *Tastenkürzel/Auf doppelte Tastenbelegung prüfen.*
2. Benutzen zwei Tastenkürzel dieselbe Tastenkombination, gibt der Tastenkürzeleditor die Meldung »Duplicate key value found« aus und markiert das zweite Tastenkürzel. Beheben Sie diesen Fehler und durchsuchen Sie die Tastenkrzeltabelle nach Doppelbelegungen solange, bis »No duplicate key values found« gemeldet wird.»

Erstellung eines Beispiels für eine Tastenkürzeltabelle

In diesem Abschnitt erstellen Sie eine Tastenkürzeltabelle für das Widgets-Menü. Ohne den Resource Workshop hätten Sie einen Texteditor oder ein Textverarbeitungsprogramm gebraucht, um das Ressourcen Script zu erstellen.

Haben Sie das Beispiel nicht gespeichert, können Sie denoch das folgende Beispiel nachvollziehen. Haben Sie das Menü gespeichert, haben einige Schritte zusätzliche oder andere Informationen für Sie.

Um eine Tastenkürzeltabelle mit dem Tastenkürzeleditor zu erstellen,

Kapitel 14 beschreibt, wie Sie ein Projekt öffnen.

1. Öffnen Sie eine PD, ggf. das Projekt mit dem Widget Menü.
2. Wählen Sie *Ressource/Neu*
3. Im Dialogfenster Neue Ressource doppelklicken Sie auf den

 ACCELERATOR Ressourcentyp. Der Tastenkürzeleditor erscheint mit einem Eintrag.

 Haben Sie das Beispielmenü gespeichert, doppelklicken Sie auf dessen Namen im Projektfenster, um den Menüeditor aufzurufen. Skalieren Sie die Größe der beiden Editoren so, daß Sie beide gleichzeitig einsehen können. Im Scriptfenster des Menüeditors markieren Sie List menu item und merken Sie sich dessen Bezeichner wmnu_List und das vorgesehene Tastenkürzel [*Strg*]+[*L*].

 Klicken Sie auf das neue Tastenkürzel im Tastenkürzeleditor. Wenn gewünscht, können Sie mit [*F6*] das Eingabefeld *Befehl* hervorheben.

4. Im Eingabefeld *Befehl* geben Sie den Namen des Bezeichners für das erste Kommando im Menü ein (wmnu_List).

5. Wechseln Sie mit [Tab] zum Feld Taste.
Der Tastenkürzeleditor befindet sich nun im Tastenwert-Modus. Drücken Sie [*Strg*]+[*L*]. Der Tastenkürzeleditor übernimmt ^L als ASCII-Taste und setzt die ASCII Schaltfläche für Sie.

6. Mit [*Alt*]+[*Esc*] verläßt der Editor den Tastenwert-Modus. Drücken Sie die Eingabetaste, so daß die Veränderungen ab sofort für das markierte Tastenkürzel im Script-Ausschnitt gelten.
Wenn Sie nicht mit einem existierenden Projekt arbeiten, fragt der Resource Workshop Sie, ob Sie einene Bezeichner für dieses Tastenkürzel erzeugen wollen. Klicken Sie auf *Ja* und dann *OK* im Dialogfenster für den neuen Bezeichner, um den Vorgabewert zu akzeptieren.

7. Mit [*Einfg*] erzeugen Sie ein neues Tastenkürzel.
Arbeiten Sie mit dem Menü Widget aus Kapitel 16, klicken Sie in den Menüeditor, markieren Sie den nächsten Befehl und merken Sie sich dessen Bezeichner und das beabsichtigte Tastenkürzel.

8. Fügen Sie die verbliebenen Tastenkürzel *-wmnu_Add*, *wmnu_Asc* und *wmnu_Desc* - hinzu und verknüpfen Sie sie mit den dazugehörigen Bezeichnern und Tastenkombinationen, bis alle Tastenkürzel definiert sind.

9. Speichern Sie das Projekt

Die Tastenkürzeltabelle ist fertig. In den folgenden Schritten wird eine Tastenkombination absichtlich doppelt zugewiesen, so daß Sie sehen können, wie die Fehlersuche mit dem Tastenkürzeleditor funktioniert.

1. Markieren Sie das zweite Tastenkürzel und ändern Sie den Taste Wert von [*Strg*]+[*A*] auf [*Strg*]+[*L*].
Mit [*Strg*]+[*Oben*] markieren Sie das Tastenkürzel, und mit [*Tab*] kommt der Tastenkürzeleditor in den Tastenwert-Modus.

2. Drücken Sie [*Strg*]+[*L*], dann [*Alt*]+[*Esc*] und schließlich die Eingabetaste, damit die Änderungen Gültigkeit erlangen.

3. Wählen Sie *Tastenkürzel/Auf doppelte Tastenbelegung prüfen*. Das Fenster Doppelte Tastenbelegung gefunden erscheint. Schließen Sie es, indem Sie auf die Eingabetaste drücken.

4. Markieren Sie im Scriptabschnitt nun wmnu_Auf oder wmnu_Abst und drücken Sie auf [*Entf*].

5. Öffnen Sie das Menü *Bearbeiten*, das die Möglichkeit enthält, mit Rückgängig Eintrag Löschen das soeben gelöschte Tastenkürzel wiederherzustellen.

6. Stelllen Sie das Element mit Rückgängig Eintrag löschen wieder her.
7. Öffnen Sie erneut das Menü *Bearbeiten*. Neben dem Befehl *Rückgängig* sehen Sie nun *Eintrag ändern*. Wird der Befehl ausgeführt, so wird das Tastenkürzel von [*Strg*]+[*L*] auf [*Strg*]+[*A*] zurückgesetzt.
8. Probieren Sie erneut *Tastenkürzel/Auf doppelte Tastenbelegung prüfen*. Jetzt sollte die Meldung »No dupplicate key values« angezeigt werden.

Dieses Beispiel zeigt, wie einfach das Anlegen von Tastenkürzeln im Zusammenspiel von Menü und Tastenkürzeleditor ist. Sie können zwischen den beiden Editoren wechseln, um festzustellen, welche Tastenkürzel welchen Befehlen zugeordnet sind. Wenn Sie Tasten im Tastenwert-Modus eingeben, übernimmt der Tastenkürzeleditor die meiste Arbeit. Sobald Sie fertig sind, sollten Sie prüfen, ob Sie doppelte Tastenkürzel eingegeben haben. Falls ja, können Sie die nötigen Änderungen sowohl im Menü als auch in der Tastenkürzeltabelle leicht vornehmen.

KAPITEL 18

Erzeugen einer String-Tabelle

Eine *String-Tabelle* enthält Fehlermeldungen, Eingabeaufforderungen oder jegliche andere Zeichenketten (Strings), die Ihre Anwendung anzeigen muß. Sie können mehrere String-Tabellen in Ihrer Projektdatei speichern. Normalerweise werden Sie eine eigene String-Tabelle für jede logisch zusammenhängende Gruppierung in Ihrem Programm definieren.

Die Definition von Textstrings als getrennte Ressource macht es einfacher, den Text zu bearbeiten, ohne daß der Quellcode geändert werden muß. Wenn Sie zum Beispiel eine Windows-Anwendung in eine Fremdsprache übersetzen, ist die Übersetzung erheblich einfacher, wenn der meiste Text in der String-Tabelle vorliegt. (Sie müßten denoch den Text in anderen Ressourcen wie Dialogfenstern übersetzen.)

Das Arbeiten mit String-Tabellen umfaßt vier grundlegende Aufgaben:

1. Den Aufruf des String-Editors
2. Erzeugen und Bearbeiten der String-Tabelle.
3. Speichern der String-Tabelle.
4. Compilieren der Ressource in eine ausführbare Datei und Testen der String-Tabelle.

Dieses Buches enthält einen kurzen Lehrgang, welche Schritte für das Erzeugen und Bearbeiten einer String-Tabelle unternommen werden müssen.

Der Aufruf des String-Editors

Wie Sie den Stringeditor aufrufen hängt davon ab, ob Sie eine neue String-Tabelle erzeugen oder eine bereits vorhandene Stringtabelle bearbeiten wollen.

Eine neue String-Tabelle erzeugen

Um den String-Editor aufzurufen und eine neue String-Tabelle zu erzeugen:

1. Öffnen Sie das Projekt, dem Sie eine neue String-Tabelle hinzufügen wollen.
2. Wählen Sie *Ressource/Neu*. Ressource Workshop zeigt das Dialogfenster *Neue Ressource*.
3. Suchen Sie im Ressourcen-Typ Listenfeld STRINGTABLE und doppelklicken Sie darauf. Ressource Workshop öffnet den String-Editor und erzeugt einen Verweis auf die neue String-Tabelle in Ihrem Projektfenster.

Eine vorhandene String-Tabelle bearbeiten

Um den String-Editor aufzurufen und eine vorhandene String-Tabelle zu bearbeiten

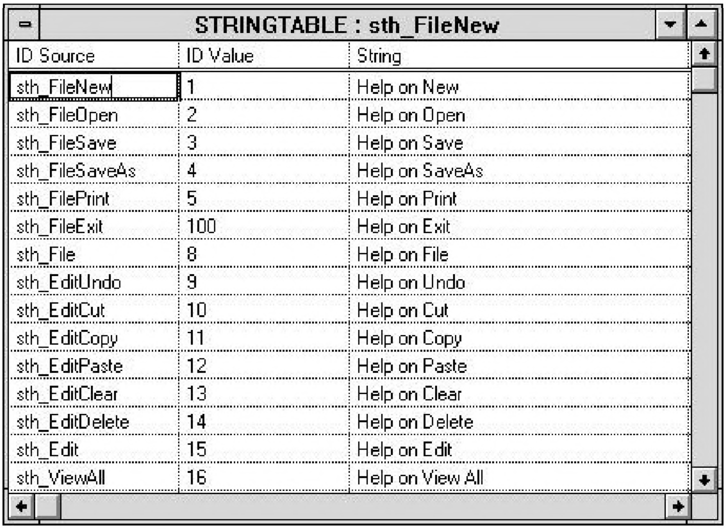
1. Öffnen Sie das Projekt, welches die String-Tabelle enthält, die Sie bearbeiten wollen.
2. Ermitteln Sie die zu bearbeitende String-Tabelle im Projektfenster.
3. Doppelklicken Sie auf diese String-Tabelle oder markieren Sie sie und wählen dann *Ressource/Bearbeiten*.

Die ausgewählte String-Tabelle erscheint im String-Editor.

Das Arbeiten mit String-Tabellen

Bei Aufruf des String-Editors erscheint dieser mit einer String-Tabelle. Wenn Sie eine neue String-Tabelle erzeugen, sehen Sie nur einen einzigen vorläufigen Eintrag: Den generischen Text »string«. Wenn Sie eine bestehende String-Tabelle bearbeiten, finden Sie Einträge, die in etwa so aussehen:

Abbildung 18.1: String-Editor mit String-Tabelle und Einträgen


STRINGTABLE : sth_FileNew

ID Source	ID Value	String
sth_FileNew	1	Help on New
sth_FileOpen	2	Help on Open
sth_FileSave	3	Help on Save
sth_FileSaveAs	4	Help on SaveAs
sth_FilePrint	5	Help on Print
sth_FileExit	100	Help on Exit
sth_File	8	Help on File
sth_EditUndo	9	Help on Undo
sth_EditCut	10	Help on Cut
sth_EditCopy	11	Help on Copy
sth_EditPaste	12	Help on Paste
sth_EditClear	13	Help on Clear
sth_EditDelete	14	Help on Delete
sth_Edit	15	Help on Edit
sth_ViewAll	16	Help on View All

Jeder String-Tabellen Eintrag benötigt eine ID_Quelle, einen ID_Wert, und den String selber.

- Eine ID_Quelle enthält eine Nummer für den String. Wenn Sie eine Variable als ID zuweisen, erscheint diese hier. Anderenfalls sehen Sie die ID_Nummer.
- Der ID_Wert enthält immer die ID_Nummer für den String.
- Ein String wird als Textkette mit höchstens 255 Buchstaben gespeichert.

Sie können einen Eintrag in der String-Tabelle mit der rechten Maustaste anklicken, um ein lokales Menü einzusehen, mit dem sich bestimmte Aufgaben besonders schnell ausführen lassen. (Sie können ebenso einen Eintrag markieren und [*Alt+F10*] drücken). Der nächste Abschnitt beschreibt, wie Windows mit String ID-Werten arbeitet und schlägt eine Methode vor, wie Sie Ihren Speicherplatz möglichst effizient einsetzen.

Strings unter Windows

Jeder String in einer String-Tabelle muß eindeutig durch seine ID_Nummer bestimmbar sein. Windows gruppiert Strings in Segmenten zu je 16 Strings. Strings mit ID-Nummern von 0 bis 15 bilden das erste Segment, Strings von 16 bis 31 das zweite Segment und so weiter. Wenn Sie Ihre Ressourcen compilieren lassen, werden die Strings an die ausführbare Datei in Segmenten angefügt, die während der Ausführung in den Arbeitsspeicher geladen werden.

Windows lädt das gesamte String-Segment in den Arbeitsspeicher jedesmal, wenn Ihre Anwendung einen bestimmten String benötigt. Wenn Sie

sorgfältig planen, welche Werte Sie den String-IDs zuweisen, können Sie den Speicherbedarf Ihrer Anwendung reduzieren.

Nehmen wir an, Sie definieren 32 Strings für Ihre Anwendung. Wenn Sie diesen String-IDs von 0 bis 31 zuweisen, wird Ihre ausführbare Datei zwei 16-Strin-Segmente beinhalten. Jedesmal, wenn Ihre Anwendung einen String benötigt und ein Segment lädt, werden möglicherweise eine ganze Anzahl nicht benötigter Strings mitgeladen.

Um Ihren Arbeitsspeicher besser zu nutzen, gruppieren Sie Ihre Strings nach logischen Gesichtspunkten. Zum Beispiel benötigt ein Teil Ihrer Anwendung 5 Strings, ein anderer Teil vielleicht 9 Strings. Wenn Sie die erste Gruppe 0 bis 4 numerieren und die zweite 16 bis 24, erzeugen Sie zwei Segmente, eines mit 11 und das andere mit 7 unbenutzten IDs. Jede unbenutzte ID benötigt nur ein Byte Arbeitsspeicher, also erheblich weniger als die Speichermenge, die von einem unbenutzten String benötigt wird.

Zur Angabe der sonst an keiner anderen Stelle benutzten String-ID können Sie eine Zahl oder einen alphanumerischen Bezeichner verwenden, der für eine Integer Zahl steht (das entspricht einem #define in C oder C++, oder einer Konstantendeklaration in Pascal). Wenn Sie sich entscheiden, einen alphanumerischen Bezeichner zu verwenden, damit Sie sich die String ID besser merken können, speichern Sie Ihre Bezeichner in einer Bezeichner Datei (einer Header-Datei für C bzw C++ oder einer unit- oder include-Datei in Pascal). Stellen Sie sicher, daß eine solche Datei existiert, bevor Sie Bezeichner vom String-Editor aus hinzufügen.

Neue Strings eingeben

Einfügen eines neuen String in die String-Tabelle :

- Wenn die Tabelle neu ist, tragen Sie die Informationen für den String wie in den Schritten 3,4 und 5 beschrieben ein.
- Wenn Sie einen String zu einer bestehenden Tabelle hinzufügen, beginnen Sie mit Schritt 1.

1. Markieren Sie den String über der Zeile, in die Sie einen neuen String einfügen wollen.
2. Drücken Sie [*Einfügen*] oder wählen Sie *String-Tabelle/Neuer Eintrag*.
3. Sie können die Zahl übernehmen, die der String-Editor ins Feld ID-Wert schreibt, oder Sie können eine eigene Zahl oder eine ID-Quelle für den Bezeichner eingeben.

 Wenn Sie eine Integer Zahl eingeben, trägt der String-Editor sie automatisch in das Feld ID-Wert ein.

 Wenn Sie einen Bezeichner eingeben, überprüft Ressource Workshop, ob ein solcher bereits existiert. Wenn ja, trägt der String-Editor den da-

zugehörigen Zahlenwert in das Feld ID-Wert ein, wenn Sie durch Drücken der [*Tab*]-Taste zum Feld String gehen. Wenn nicht, wird das Dialogfenster *Neuen Bezeichner* erzeugen angezeigt.

Um das Stringfeld auf die ursprünglichen Einstellungen zurückzusetzen, drücken Sie [*Esc*] und dann [*Tab*] oder die *Eingabetaste*

4. Drücken Sie [*Tab*] oder klicken Sie in das Fenster unter String und geben Sie den Textstring ein.
 Jeder String kann höchstens 255 Zeichen umfassen und darf beliebige Escape-Sequenzen wie in C enthalten, einschließlich der folgenden: \n (Zeilenwechsel), \t (Tab), \r (Eingabe bzw Return), \\ (Backslash), \ (Anführungszeichen).

Wenn der Ressource Workshop Compiler eine C-artige Escape-Sequenz in einem String entdeckt, wird der dazugehörige hexadezimale ASCII-Code in den Objektcode eingefügt. Es kommt auf Ihr Programm an, ob die Werte richtig interpretiert werden. Wenn der Compiler zum Beispiel \b\040\x7F einliest, erzeugt er die hexadezimale Sequenz 07207F. Ihr Programm könnte dies als ASCII-Zeichen BEL, SPC und DEL interpretieren, oder diesen Werten eine ganz andere Bedeutung zuschreiben.

5. Drücken Sie [*Eingabe*] (um den neuen Wert einzugeben) oder [*Einfügen*] (um den Wert einzugeben und einen neuen einzufügen).

Bearbeiten vorhandener Strings

Mit dem String-Editor ist es einfach, einzelne Strings zu verändern. Um mit der Mouse einen String zum Bearbeiten zu wählen, klicken Sie einfach darauf. Mit den Tasten [*Tab*], [*Oben*] und [*Unten*] können Sie sich in der Tabelle bewegen. Plazieren Sie Ihren Cursor auf den String, den Sie bearbeiten wollen.

Verändern eines Strings

Sie können ID-Quelle und ID-Wert für jeden String löschen und neue Werte eingeben. Sie können nicht direkt verändern, was im ID-Wertefeld angezeigt wird, aber der String-Editor bringt das Feld auf den neuesten Stand, je nachdem, was Sie im Feld ID-Quelle eingegeben haben.

Bearbeiten eines Ressourcenskripts einer String-Tabelle

Sie können den internen Texteditor zum Bearbeiten des Ressourcenskripts einer String-Tabelle verwenden. Wählen Sie dazu eine String-Tabelle im Projektfenster und fahren mit *Ressourcen/Bearbeiten* als Text fort.

Der Text des Ressourcenskripts erscheint und Sie können ihn bearbeiten.

Verändern des String

Bearbeiten eines String:

1. Suchen Sie den String und machen dort alle gewünschten Veränderungen. Verändern Sie nur den Text zwischen Anführungszeichen.

2. Um ihre Eingabe zu compilieren, ohne den String-Editor zu verlassen, wählen Sie *Compilieren/Compilieren*.

 Beachten Sie, daß der Menüpunkt Compilieren nur im Text-Editor verfügbar ist.

3. Um den Texteditor zu verlassen, wählen Sie im Steuer-Menü des Editors die Anweisung Schließen.
 Ressource Workshop fragt zurück, ob Sie kompilieren wollen. Klicken Sie auf [Ja], damit Ressource Workshop compiliert und anschließend zum Projektfenster zurückgeht.

Wird ein Syntax-Fehler gefunden, schaltet Ressource Workshop in den Texteditor zurück, so daß Sie sofort den Fehler beheben können.

Erzeugen einer Beispiel String-Tabelle

Im folgenden Beispiel wird eine String-Tabelle erzeugt, die Ressource Workshop benutzt, um Menü-Optionen zu beschreiben.

Sie würden ohne Ressource Workshop das folgende Ressourcenskript benutzen müssen, um diese Strings zu erzeugen:

Diese Strings tauchen in der Ressource Workshop Statuszeile auf

```
STRINGTABLE
BEGIN
  MENU_FILE,   "Create, open or close files"
  MI_FILENEW,  "Create a new project, ressource or file"
  MI_FILEOPEN, "Open a resource file"
  MI_FILESAVE, "Save this resource file"
END
```

Vor jedem String ist ein sonst an keiner anderen Stelle benutzter Bezeichner in Großbuchstaben angegeben. Wie bei allen Windows-Ressourcen benötigt jeder String eine Integer-IDs. Ohne Ressource Workshop müßten Sie die korrespondierenden Integer IDs für alle Bezeichner in einer Header Datei (bei einem C-Programm) oder in einer Include Datei oder in einem Unit (bei einem Pascal-Programm) definieren.

So erzeugen Sie die Beispielstrings mit dem String-Editor:

1. Stellen Sie sicher, daß ein Projekt geöffnet ist. Wenn Sie die Beispiele mit MYPROJ.RC nachvollzogen haben, können Sie ja dieses Projekt öffnen.

2. Wenn Sie nicht bereits eine Bezeichner-Datei (eine Header Datei für #defines in C oder eine Include-Datei bzw. eine Unit für Konstantendeklarationen in Pascal) für dieses Projekt erstellt haben, erstelllen Sie die Datei jetzt und nennen Sie sie MYPROJ.H oder MYPROJ.PAS

3. Wählen Sie *Ressource/Neu* und doppelklicken Sie auf STRINGTABLE. Der String-Editor wird aufgerufen.
4. Löschen Sie die Nummer im Textfenster unter ID-Quelle und geben Sie den Bezeichner für den String ein. Für den ersten String ist das MENU_FILE.
5. Drücken Sie [*TAB*], um zum Feld *String* zu kommen.

 Bevor Sie der String-Editor im String Feld arbeiten läßt, sucht er nach einer Integer-ID für den aktuellen String. Zuerst wird untersucht, was Sie unter *ID-Source* eingegeben haben. Hätten Sie eine Integernummer eingegeben, hätte der String-Editor denselben Integerwert für ID-Wert eingetragen und Sie dann im Feld String weiterarbeiten lassen.

 Da Sie aber einen alphanumerischen Bezeichner (in diesem Fall MENU_FILE) eingegeben haben, sucht der String-Editor nach einem C #define oder einer Pascal-Konstanten-Deklaration, welche auf einen Integerwert verweist. Da keine da ist, werden Sie gefragt, ob Sie einen neuen Bezeichner erzeugen wollen.
6. Klicken Sie auf Ja, um das »Neuer Bezeichner«-Dialogfenster angezeigt zu bekommen.
7. Suchen Sie im Datei-Listenfeld, bis Sie MYPROJ.H (oder MYPROJ.PAS) gefunden haben. Wenn Sie darauf doppelklicken, wählen Sie es als die Bezeichnerdatei aus, in die der neue Bezeichner eingefügt wird.
8. Geben Sie eine sonst an keiner anderen Stelle benutzte Integer-ID im Textfenster für Werte ein. Geben Sie für den ersten Bezeichner 768 ein.
9. Wenn Sie die [*Eingabetaste*] drücken oder auf OK klicken, wird der neue Bezeichner in MYPROJ.H oder MYPROJ.PAS eingefügt.
10. Nachdem der Bezeichner eingegeben ist, befinden Sie sich wieder im String-Feld. Geben Sie nun den Text für den String ein. Für den ersten String ist das *create, open, or close files*.
11. Um den nächsten Textstring zu definieren, drücken Sie [*Einfügen*]oder wählen *Stringtabelle/Neuer Eintrag*.

Wiederholen Sie nun die Schritte 4 -11, um die weiteren drei Strings einzugeben, die zu Beginn dieses Abschnitts angegeben worden sind: MI_FILENEW, MI_FILEOPEN und MY_FILESAVE.

Es wird Ihnen auffallen, daß der Stringeditor für jede neue Integer-ID zuletzt eingegebenen Integer-ID-Wert um 1 erhöht. Sie können diese Nummer einfach übernehmen oder andere Integerwerte eintragen.

Nachdem alle vier Strings erzeugt wurden, sollte Ihre String-Tabelle wie folgt aussehen:

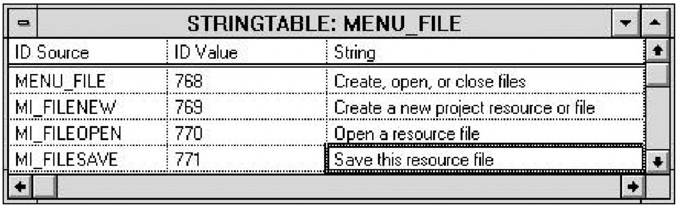


Abbildung 18.2: String-Editor mit vier definierten Strings

Zu guter letzt schließen Sie die String-Tabelle, indem Sie Schließen aus dem Steuer-Menü des String-Editors wählen. Der String-Editor benennt die String-Tabelle nach dem Namen des ersten Bezeichners in der Tabelle. Wäre der erste ID-Quelle-Eintrag einen Nummer gewesen, wäre diese Nummer der Name der neuen String-Tabelle. Der Name einer String-Tabelle kann nur geändert werden, indem der erste ID-Quelleneintrag der Tabelle geändert wird.

Diese Art der Namensvergabe ist sinnvoll, wenn Sie bedenken, daß Strings in Segmenten zu 16 Strings geladen werden und die Integer-ID angibt, wo der String im Segment zu finden ist. Die Integer-ID des ersten Strings im Segment zeigt also an, wo die Tabelle im Segment beginnt.

K A P I T E L 19

Anwendung des Paint-Editors

Der Paint-Editor ist ein Werkzeug, mit dem Sie jede Bitmap-Ressource erzeugen und bearbeiten können. Dies umfaßt auch die Standard-Bitmap-Ressourcen:

- Symbole
- Mauszeiger
- Bitmaps
- Schriften

Eigenschaften, die nur bestimmten Bitmap-Ressourcen eigen sind, werden in den Kapiteln 20 bis 22 beschrieben.

Dieses Kapitel beschreibt die Eigenschaften und die Werkzeuge des Paint-Editors, die im allgemeinen auf alle Bitmap-Ressourcen angewendet werden können. Die Kapitel über die jeweiligen Ressourcen beschreiben diejenigen Eigenschaften und Werkzeuge des Paint-Editors, die nur diesen Ressourcen zu eigen ist.

Aufruf des Paint-Editors

Wenn Sie eine Bitmap-Ressource bearbeiten oder eine neue erzeugen wollen, wird der Paint-Editor automatisch aufgerufen. Die besonderen Schritte, die zum Aufruf des Paint-Editors notwendig sind, werden in den Kapiteln über die einzelnen Bitmap-Ressourcen beschrieben.

Wenn Sie eine neue Bitmap-Ressource erzeugen wollen und es ist bereits ein Bitmap in der Zwischenablage vorhanden, wird die Größe des Bitmap in der Zwischenablage als Größe Ihres neuen Bitmap in das Dialogfenster »Neue-Bitmap-Attribute« übernommen. Befindet sich kein Bitmap in der Zwischenablage, wird als Größe standardmäßig 64x64 Pixel angenommen.

Pixel, Vordergrund- und Hintergrundfarben

Weiter hinten in diesem Kapitel finden Sie mehr Informationen über Vorder- und Hintergrundfarben.

Die mit dem Paint-Editor erzeugten Bitmap-Zeichnungen werden durch Einfärben von annähernd rechteckigen »Punkten« (*Pixel*) erzeugt, die in Form eines Gitters gegeben sind. Jedem Pixel kann eine sogenannte *Vorder-* oder *Hintergrundfarbe* zugewiesen werden. Die Vordergrundfarbe wählen und zeichnen Sie mit der linken Maustaste, die Hintergrundfarbe mit der rechten Maustaste.

Sie können eine Vielzahl von Vorder- und Hintergrundfarben sowohl für die Formen Ihrer Zeichnung (Linien, Rechtecke, Schatten usw.) als auch für den Hintergrund benutzen. Durch die Möglichkeit, beiden Maustasten Farben zuzuweisen, stehen Ihnen immer gleichzeitig zwei Farben zur Verfügung.

➥ Es gibt einen wichtigen Unterschied zwischen Vorder- und Hintergrundfarbe. Wenn Sie einen Block von Pixeln löschen oder ersetzen, ersetzt die aktuell gewählte Hintergrundfarbe die bisherige Einfärbung der Pixel.

Wenn Sie den Radierer benutzen, werden die Verhältnisse umgekehrt. Mit der linken Maustaste erreichen Sie die Hintergrundfarbe, mit der rechten Maustaste die Vordergrundfarbe.

Die aktuelle Vordergrundfarbe wird durch die Buchstaben *VG* in der Farbpalette angezeigt, die Hintergrundfarbe mit *HG*. Sie können diese Einstellung jederzeit ändern. Wählen Sie für Vorder- und Hintergrund dieselbe Farbe, taucht an entsprechender Stelle in der Palette nur *VH* auf.

Anwenden der Werkzeugpalette

Wenn Sie eine Ressource in den Paint-Editor laden, wird die Werkzeugpalette in der oberen rechten Ecke des Fensters angezeigt. Sie können aus der Palette dasjenige Werkzeug auswählen, mit dem Sie arbeiten wollen. Am unteren Ende der Palette finden Sie die Auswahlmöglichkeit für Pinselformen, Füllmuster und Linienbreite.

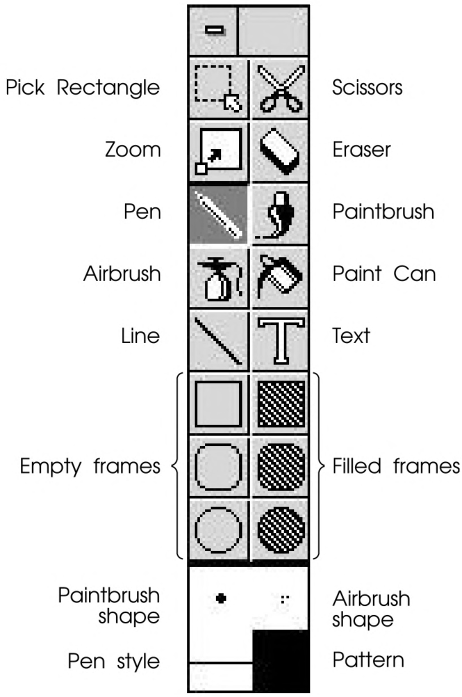


Abbildung 19.1: Die Werkzeugpalette des Paint-Editors

Die Werkzeugpalette hat ähnliche Eigenschaften wie ein Fenster: Sie können sie bewegen, schließen und öffnen.

Die meisten Werkzeuge lassen Sie entweder mit der Vorder- oder Hintergrundfarbe zeichnen.

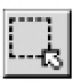
Wenn Sie nur VG in der Farbpalette sehen, ist dieselbe Farbe als Vorder- und Hintergrundfarbe gewählt.

- Benutzen Sie die linke Maustaste, um mit der aktuellen Vordergrundfarbe zu zeichnen (*VG* in der Farbenpalette).
- Benutzen Sie die rechte Maustaste, um mit der aktuellen Hintergrundfarbe zu zeichnen (*HG* in der Farbenpalette).

Um ein Werkzeug aus der Werkzeugpalette zu wählen, klicken Sie darauf. Die folgenden Abschnitte beschreiben jedes einzelne Werkzeug.

Markierungsrahmen setzen

Um die gesamte Zeichnung auszuwählen, benutzen Sie *Bearbeiten/Alles markieren*.



Mit dem Werkzeug Markierungsrahmen setzen wird ein rechteckiger Bereich Ihrer Zeichnung für das Bewegen, Kopieren oder Löschen ausgewählt. Um den Markierungsrahmen zu erzeugen, positionieren Sie die Spitze des Pfeils an die Stelle, wo Sie eine Ecke des Markierungsrahmens setzen wollen, und ziehen Sie dann den Markierungsrahmen in die gewünschte Größe. Umfaßt der Rahmen den Bereich, den Sie auswählen wollen, lassen Sie die Maustaste los. Um den markierten Bereich zu löschen, klicken Sie entweder außerhalb des Bereichs, oder drücken die [*Eingabe*]- oder [*Esc*]-Taste.

Nachdem Sie einen Bereich gewählt haben, können Sie den gewählten Bereich mit den Befehlen des Bearbeiten-Menüs ausschneiden, kopieren, löschen, duplizieren oder einfügen oder mit den Bereich der Maus bewegen oder duplizieren.

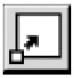
Schere



Die Schere hat grundsätzlich dieselbe Funktion wie das Werkzeug zum Setzen eines Markierungsrahmens, jedoch können Sie hiermit einen Bereich beliebiger Form auswählen.

Um einen Bereich zu markieren, ziehen Sie mit dem Scherenzeiger, bis der gewünschte Bereich umfahren wurde, und lassen dann die Maustaste los. Der gewählte Bereich wird durch ein blinkendes Muster angezeigt. Sie können den Bereich genauso Kopieren, Bewegen oder Löschen, wie Sie es mit einem Bereich tun würden, der von einem Markierungsrahmen umgeben ist.

Vergrößern/Verkleinern



Der Paint-Editor benutzt beim Vergrößern das Zentrum des Bildes als Bezugspunkt .

Das Werkzeug Vergrößern erlaubt Ihnen, das gesamte Bild oder auch nur einen Ausschnitt davon zu vergrößern oder zu verkleinern.

Um das gesamte Bild zu vergrößern, doppelklicken Sie auf das Vergrößern-Werkzeug. Das Bild wird zur nächstgrößeren Darstellung vergrößert: 400%, 800% oder 1600%. Sie können auch *Ansicht/Vergrößern* benutzen, um denselben Effekt zu erzielen.

Wenn Sie das Bild vergrößert haben, benutzen Sie das Hand-Werkzeug (siehe weiter hinten in diesem Kapitel) oder die Bildlaufleisten, um sich im vergrößerten Bild zu bewegen.

Um das Bild wieder zu verkleinern, halten Sie die [*Umschalt*]-Taste gedrückt und doppelklicken auf das Vergrößern-Werkzeug. Sie können auch Ansicht/Verkleinern benutzen, um denselben Effekt zu erzielen. Um nur einen Ausschnitt des Bildes zu vergrößern, markieren Sie den Ausschnitt durch Ziehen mit dem Vergrößern-Werkzeug, und lassen Sie dann die Maustaste los. Der Ausschnitt wird zur nächstgrößeren Darstellung vergrößert: 100%, 400% oder 800%.

Tabelle 19.1: Vergrößern-Befehle

Ansicht-Befehl	Tastenkürzel	Mausaktion am Vergrößern-Symbol
Vergrößern	[*Strg*]+[*Z*]	Doppelklicken
Verkleinern	[*Strg*]+[*O*]	[*Umschalt*]+doppelklicken
Reale Größe	[*Strg*]+[*A*]	Keine

Um die Bilder pixelweise zu bearbeiten, können Sie sich bei jedem vergrößerten Bild ein Gitter anzeigen lassen. Das wird angezeigt mit *Optionen/Editoroptionen* und klicken auf den Befehl *Gitter auf vergrößerte Fenster*.

Wenn Sie mit zwei Fensterausschnitten arbeiten, betrifft das Vergrößern nur das aktuell aktive Fenster.

Radierer



Der Radierer kann zum Löschen des gesamten Bildes oder zum Zeichnen verwendet werden. Beachten Sie, daß die Belegung der Maustasten beim Radierer genau umgekehrt ist wie bei den anderen Zeichenwerkzeugen.

- Wenn Sie auf den Radierer in der Werkzeugpalette doppelklicken, wird das gesamte Bild in der aktuellen Hintergrundfarbe überfärbt.
- Ziehen Sie mit der linken Maustaste, zeichnet der Radierer eine Linie von einem Pixel Breite in der aktuellen Hintergrundfarbe.
- Ziehen Sie mit der rechten Maustaste, zeichnet der Radierer eine Linie von einem Pixel Breite in der aktuellen Vordergrundfarbe (*VG* in der Farbpalette).

Wenn Sie nur *VG* in der Farbenpalette sehen, ist dieselbe Farbe als Vorder- und Hintergrundfarbe gewählt.

Bevor Sie den Radierer benutzen, sollten Sie die aktuelle Farbauswahl in der Farbenpalette überprüfen.

Stift



Das Stift-Werkzeug erlaubt Ihnen die Anfertigung von Freihand-Zeichnungen (siehe Abb. 19.1). Um mit dem Stift zu zeichnen, drücken Sie eine Maustaste und ziehen Sie mit dem Mauszeiger über das Bild. Wenn Sie mit dem Zeichnen fertig sind, lassen Sie die Maustaste los. (Um absolut gerade Linien zu zeichnen, sollten Sie stattdessen das Linie-Werkzeug aus der Werkzeugpalette benutzen.)

Pinsel



Der Pinsel erstellt Freihandzeichnungen in der aktuellen Pinselform mit dem aktuell gewählten Hintergrundmuster (siehe Abb. 19.1). Um zu zeichnen, ziehen Sie mit dem Mauszeiger über das Bild. Wenn Sie mit dem Zeichnen fertig sind, lassen Sie die Maustaste los.

Wenn Sie das Werkzeug Pinsel markieren, nimmt der Mauszeiger das Aussehen der ausgewählten Pinselform an. Der Bereich, der vom Pinsel gezeichnet wird, ist immer im Verhältnis zur Größe des Bildes gleich. Das heißt, ist der Pinselstrich halb so groß wie die Bildbreite, und Sie vergrößern das Bild, so ist der Pinselstrich immer noch halb so groß wie die Breite des jetzt vergrößerten Bildes.

Bevor Sie den Pinsel benutzen, geben Sie die Pinselform, das Füllmuster und die Farben an.

Airbrush



Das Airbrush-Werkzeug erstellt Freihandzeichnungen mit den aktuell gewählten Airbrush-Düsen und den Füllmustern aus der Werkzeugpalette. Der Airbrush kann auf zweierlei Arten verwendet werden:

- Sie können mit dem Airbrush über das Bild ziehen. Vom Pinsel unterscheidet sich der Airbrush insofern, als der Airbrush ein dickeres Muster erstellt, wenn Sie ihn langsam über das Bild ziehen, und ein dünneres Muster, wenn Sie ihn schnell übers Bild ziehen.

- Sie können wiederholt klicken, so als ob Sie eine Spraydose betätigen.

Wenn Sie den Airbrush markieren, nimmt der Mauszeiger das Aussehen der ausgewählten Düse an. Der Bereich, der vom Airbrush bestrichen wird, ist immer im Verhältnis zur Größe des Bildes gleich. Das heißt, ist der Airbrush-Strich halb so groß wie die Bildbreite, und Sie vergrößern das Bild, so ist der Airbrush-Strich immer noch halb so groß wie die Breite des jetzt vergrößerten Bildes.

Bevor Sie den Airbrush benutzen, sollten Sie die Airbrush-Düse, das Füllmuster und die Farben bestimmen.

Farbeimer



Das Werkzeug Farbeimer füllt einen Bereich Ihres Bildes mit der ausgewählten Farbe aus. Um den Farbeimer zu benutzen, positionieren Sie den Farbeimer-Mauszeiger (in Form eines Fadenkreuzes) in den Bereich des Bildes, den Sie einfärben wollen, und klicken sodann. Der Farbeimer ersetzt die ursprüngliche Farbe unter dem Fadenkreuz mit Ihrer ausgewählten Farbe und färbt die Umgebung weiter ein, bis er an Pixel in einer anderen Farbe als der ursprünglichen stößt.

Ist zum Beispiel die gewählte Farbe Rot, und Sie klicken in ein blaues Rechteck, wird jedes Blau in der Umgebung durch Rot ersetzt. Der Farbeimer wird keine andere Farbe außer Blau ersetzen. Ist das blaue Rechteck von einem grünen Rahmen umgeben, wird nur das blaue Rechteckinnere durch Rot ersetzt.

Wenn Sie in einen nicht gänzlich von anderen Farben umgebenen Bereich klicken, wird sich die Farbe aus dem Bereich heraus in alle anderen gleichfarbigen Bereiche ergießen.

Wegen einiger Probleme, die mit den Bildschirmtreibern zusammenhängen, funktioniert das Fluten von Bitmap-Zeichnungen mit dem Farbeimer nicht immer richtig. Um diese Probleme zu lösen, verwendet Ressource Workshop noch einen Algorithmus zum Ausgießen mit Farbe, der langsamer, aber verläßlicher arbeitet. Um diesen Algorithmus bei einem der Paint-Editoren zu verwenden, fügen Sie die folgende Zeile in den [RWS_Icon] Abschnitt in WORKSHOP.INI ein:

```
RWS_OwnFloodFill=1
```

Das folgende Beispiel zeigt einen Ausschnitt aus der bearbeiteten RWS_Icon Abschnitt:

```
[RWS_Icon]
RWS_OwnFloodFill=1
PercentLeft=69
ZoomLeft=8
ZoomRight=1
bVert=1
```

Linie

Das Linie-Werkzeug zeichnet Geraden. Um zu zeichnen, ziehen Sie mit dem Mauszeiger über das Bild. Wenn Sie mit dem Zeichnen der Geraden fertig sind, lassen Sie die Maustaste los.

Für freihändig gezeichnete Linien benutzen Sie den Stift.

Wollen Sie die Linie auf 45-Grad-Schritte begrenzen, also nur waagerechte, senkrechte oder diagonale Linien zeichnen, halten Sie die [*Umschalt*]-Taste während des Zeichnens gedrückt.



Bevor Sie das Linie-Werkzeug benutzen, können Sie die Strichstärke und die Farbe angeben.

Text



Um Text in Ihre Zeichnung einzufügen, markieren Sie das Text-Werkzeug und klicken in der Zeichnung an die Stelle, wo der Text beginnen soll. Eine blinkende Eingabeaufforderung zeigt, daß Sie mit der Eingabe Ihres Textes beginnen können.

Um festzulegen, wie und wo der Text dargestellt wird,

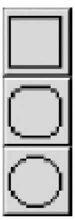
- benutzen Sie Text/Schrift , um Schriftname, Größe und Attribute Ihres Textes zu bestimmen,
- benutzen Sie Text/Linksbündig,Zentriert oder Rechtsbündig, um die Ausrichtung des Textes zu bestimmen.

Für mehr Informationen über die Befehle im Textmenü siehe »Text einer Ressource hinzufügen« weiter hinten in diesem Kapitel.

Sie müssen keinen Menübefehl ausführen, bevor Sie einen Text eingeben. Sie können die Befehle auch unmittelbar nach der Texteingabe auswählen (und bevor Sie wieder mit der Maus klicken). Wenn Sie zum Beispiel während des Tippens bemerken, daß Ihr Text zu groß ist, um ins Bild zu passen, können Sie die Eingabe unterbrechen und den Text mit dem Befehl Schrift verkleinern.

Text wird immer in der aktuellen Vordergrundfarbe dargestellt. Bevor Sie Text eingeben, können Sie also durch Anklicken einer Farbe aus der Farbpalette (mit der linken Maustaste) die Textfarbe bestimmen. Genauso wie bei Schriftart (Schriftname) und Größe können Sie auch die Farbe unmittelbar nach der Texteingabe bestimmen.

Leere Rahmen zeichnen



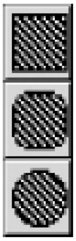
Mit einigen Werkzeugen können Sie leere Rahmen in verschiedenen Formen zeichnen: Rechteck, abgerundetes Rechteck und Ellipse.

Um einen leeren Rahmen zu zeichnen, markieren Sie das Symbol für die gewünschte Rahmenform, und ziehen Sie den Rahmen im Bild. Positionieren Sie den Fadenkreuz-Mauszeiger in eine Ecke des Rahmens, und ziehen Sie bis in die gegenüberliegende Ecke. Lassen Sie den Mauszeiger los, sobald der Rahmen das von Ihnen gewünschte Ausmaß hat. Drücken

Sie beim Ziehen die [*Umschalt*]-Taste, um ein Quadrat (bei Rechteck-Werkzeug) oder einen Kreis (bei Ellipse-Werkzeug) zu zeichnen.

Bevor Sie einen Rahmen zeichnen, können Sie die Rahmenliniendicke und die Farbe bestimmen.

Ausgefüllte Flächen zeichnen



Mit drei Werkzeugen können Sie ausgefüllte Flächen in verschiedenen Formen zeichnen: ausgefülltes Rechteck, abgerundetes ausgefülltes Rechteck und ausgefüllte Ellipse.

Um einen ausgefüllten Rahmen zu zeichnen, markieren Sie das Symbol für die gewünschte Rahmenform, und ziehen Sie den Rahmen im Bild. Positionieren Sie den Fadenkreuz-Mauszeiger in einer Ecke des Rahmens, und ziehen Sie bis in die gegenüberliegende Ecke. Lassen Sie den Mauszeiger los, sobald der Rahmen das von Ihnen gewünschte Ausmaß hat.

Für den Umriß wird die aktuelle Linienbreite benutzt. Geben Sie als Strichstärke Null an, wenn Sie nicht wollen, daß ein Umriß gezeichnet wird.

Das aktuelle Füllmuster wird in der rechten unteren Ecke der Werkzeugpalette dargestellt. Wenn Sie mit der linken Maustaste ziehen, werden das Muster und der Rahmen in der aktuell gewählten Vordergrundfarbe und der Musterhintergrund in der aktuell gewählten Hintergrundfarbe dargestellt. Ziehen Sie mit der rechten Maustaste, ist der Effekt genau umgekehrt.

Das Dialogfenster Muster bestimmen enthält zudem zwei einfarbige »Muster«, nämlich ein weißes und ein schwarzes.

- Wenn Sie das schwarze einfarbige Muster wählen, wird mit der linken Maustaste der Umriß in der Hintergrundfarbe und die Füllung in der Vordergrundfarbe gezeichnet. (Wie gewohnt, erzeugt die rechte Maustaste den umgekehrten Effekt.)
- Wenn Sie das weiße einfarbige Muster wählen, wird mit der linken Maustaste der Umriß in der Vordergrundfarbe und die Füllung in der Hintergrundfarbe gezeichnet. (Und wiederum erzeugt die rechte Maustaste den genau umgekehrten Effekt).

Bevor Sie den gefüllten Rahmen zeichnen, stehen Ihnen verschiedene Strichstärken, Farben und Füllmuster zur Auswahl.

Das Hand-Werkzeug



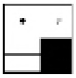
Manchmal ist es in einem vergrößerten Bild nicht möglich, alles davon im Fenster darzustellen. Sie können dann die Hand benutzen, um das Bild zu bewegen und so zu den Teilen des Bildes zu kommen, die nicht sicht-

Sie können auch die Bildlaufleiste benutzen, um das Bild zu bewegen.

bar waren. Die Hand ist nicht wie die anderen Werkzeuge über die Werkzeugpalette zu erreichen, aber Sie können vorübergehend jedes Werkzeug (mit Ausnahme des Text-Werkzeugs) in die Hand verwandeln, wenn Sie [*Strg*] drücken.

Mit der Hand können Sie das Bild greifen und in jede beliebige Richtung ziehen.

Stilauswahl



Am unteren Ende der Werkzeugpalette befinden sich, im Uhrzeigersinn gesehen, die Möglichkeiten zur Auswahl einer Pinselform, der Airbrush-Düse, des aktuellen Füllmusters und der Linienstärke.

Sie können auf eines der Symbole klicken, um eines der Attribute zu ändern.

Arbeiten mit zwei Fensterausschnitten

Im Paint-Editor können Sie zwei Ansichten Ihres Bildes einsehen, während Sie es erzeugen oder bearbeiten. Sie können Ihr Bild in der Senkrechten oder Waagerechten teilen, so daß die Ansichten nebeneinander oder übereinander zu sehen sind. Sie können sich auch dafür entscheiden, eine oder beide Ansichten zu vergrößern oder zu verkleinern.

Mit Ansicht/Horizontal teilen oder Ansicht/Vertikal teilen können Sie das Fenster teilen.

Ist das Fenster geteilt, wird einer der Ausschnitte als *aktiver* Ausschnitt angesehen. Der aktive Ausschnitt ist derjenige, mit dem Sie arbeiten. Um einen Ausschnitt zum aktiven Ausschnitt zu machen, klicken Sie darin.

Um einen Ausschnitt gegenüber dem anderen zu vergrößern, positionieren Sie den Mauszeiger auf die Trennungslinie zwischen den Ausschnitten. Sobald der Mauszeiger ein Doppelpfeil wird, können Sie an der Trennungslinie ziehen. Bei einem vertikal geteilten Fenster beispielsweise könnten Sie die Trennungslinie nach rechts ziehen, um mehr von einem vergrößerten Bild im linken Ausschnitt zu sehen.

Um die Aufteilung des Bildes rückgängig zu machen und somit zu einer einzigen Ansicht der Zeichnung zurückzukehren, ziehen Sie die Trennungslinie ganz nach oben oder nach unten, wenn es sich um eine Teilung in der Waagerechten handelt. Wurde das Fenster vertikal geteilt, ziehen Sie die Trennungslinie ganz nach links oder nach rechts.

Die Statuszeile lesen

Die Statuszeile am unteren Ende des Paint-Editors ist in zwei Teile geteilt: Die rechte Seite vermittelt aktuelle Informationen über das Werkzeug, und die linke Seite zeigt Informationen über die Menübefehle des Paint-Editors an.

Sobald Sie ein Menü anklicken oder ein Tastenkürzel benutzen, um einen Menübefehl ausführen zu lassen, zeigt die linke Seite der Statuszeile weitere Informationen über den markierten Befehl an.

Die rechte Seite der Statuszeile zeigt an, welches Zeichenwerkzeug Sie benutzen und an welcher Stelle es sich gerade befindet. Es können auch Informationen über die Farben angegeben sein, doch dies ist abhängig vom gewählten Werkzeug.

Die Meldung auf der Statuszeile hängt also davon ab, welches Werkzeug Sie markiert haben und wo sich Ihr Mauszeiger auf dem Bildschirm befindet. Im weiteren werden zwei Beispiele von Statuszeilenmeldungen erläutert.

Sie können den Palettenindex und die RGB-Einstellungen für eine Farbe in der Farbenpalette sehen, wenn Sie auf die Farbe doppelklicken.

Linie x: 18 y: 32
Die Meldung besagt, daß Sie das Linie-Werkzeug markiert haben und zu den Pixelkoordinaten 18,32 gegangen sind. Pixelkoordinaten werden von der linken oberen Ecke des Bildes aus gerechnet. Wenn Sie sich zum rechten Rand Ihres Bildes bewegen, wächst der X-Wert. Gehen Sie zum unteren Rand des Bildes, wird der Y-Wert erhöht.

Weiter hinten in diesem Kapitel finden Sie mehr Informationen über RGB-Werte und Palettenindizes.

Pinsel x: 20 y: 37 R: 128 G: 0 B: 0 Palettenindex: 1
Diese Meldung besagt, daß Sie das Pinsel-Werkzeug markiert haben und zu den Pixelkoordinaten 20,37 gegangen sind. Die Werte R, G und B zeigen die Farbwerte der Farben Rot, Grün und Blau. Die Farbe des Bildes auf 20,37 ist gegenwärtig die Farbe 1 im Palettenindex, welches einen Wert von 128 Rot, 0 Grün und 0 Blau hat.

Sie bekommen den Palettenindex und die RGB-Werte an der Stelle des Mauszeigers ebenfalls angezeigt,wenn Sie den Stift, den Airbrush oder den Farbeimer markieren.

Arbeiten mit Farben

Mit der Farbpalette können Sie Farben zur Bearbeitung Ihrer Ressource auswählen. Sie können sogar mit der Farbpalette arbeiten, wenn Ihr Bild Schwarz/Weiß ist. Sie können die Farbpalette jederzeit verbergen oder sich anzeigen lassen.

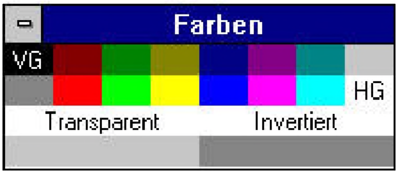


Abbildung 19.2: Farbenpalette mit 16 Farben

Sie können die Farbenpalette benutzen, um:

- eine Vordergrundfarbe zu wählen,
- eine Hintergrundfarbe zu wählen,
- transparente und invertierte Bereiche zu wählen (nur bei Mauszeiger oder Symbol-Ressourcen).

Die Anzahl der Farben für eine Ressource wählen

Wenn Sie ein neues Bitmap oder ein Symbol erzeugen, wird das Dialogfenster Neue Bitmap-Attribute angezeigt, in dem Sie die Anzahl der Farben Ihrer Ressource wählen können.

Wenn Sie ein Bitmap oder ein Symbol bearbeiten, können Sie die Anzahl der Farben mit dem Befehl Größe und Attribute ändern. Der Befehl befindet sich im Menü Bitmap oder dem Menü Symbol, je nachdem, ob Sie eine Bitmap- oder eine Symbol-Ressource bearbeiten.

Manche der Gerätetreiber für höhere Auflösungen unter Windows 3.1 benötigen 512KB Arbeitsspeicher. Für Bitmap- und Symbol-Ressourcen können Sie bis zu 256 Farben verwenden. Die Anzahl der Farben, die Sie verwenden und in der Farbpalette sehen können, hängt vom Gerätetreiber ab, den Sie mit Windows benutzen.

Vorder- und Hintergrundfarben benutzen

Benutzen einer Vordergrundfarbe:

1. Klicken Sie mit der *linken* Maustaste auf die gewünschte Farbe in der Farbpalette. Die Buchstaben *VG* erscheinen auf der angeklickten Farbe.
2. Markieren Sie ein Werkzeug zum Zeichnen oder Malen, und klicken oder ziehen Sie mit der *rechten* Maustaste, um mit der Vordergrundfarbe zu malen oder zu zeichnen.

Wenn Sie nur *VH* in der Farbpalette sehen, ist dieselbe Farbe als Vorder und Hintergrundfarbe gewählt.

Benutzen einer Hintergrundfarbe:

1. Klicken Sie mit der *rechten* Maustaste auf die gewünschte Farbe in der Farbpalette. Die Buchstaben *HG* erscheinen auf der angeklickten Farbe.
2. Markieren Sie ein Werkzeug zum Zeichnen oder Malen, und klicken oder ziehen Sie mit der rechten Maustaste, um mit der Vordergrundfarbe zu malen oder zu zeichnen.

Der Radierer funktioniert umgekehrt wie die Zeichen- oder Malwerkzeuge. Ziehen Sie mit der linken Maustaste, zeichnet der Radierer in der aktuellen Hintergrundfarbe. Ziehen Sie mit der rechten Maustaste, zeichnet der Radierer in der aktuellen Vordergrundfarbe.

Transparente und invertierte Farben

Transparente oder *invertierte* Farbbereiche können nur in Symbol- oder Mauszeiger-Ressourcen verwendet werden.

- Ein transparenter Bereich ist während der Laufzeit »durchsichtig«, so daß der Desktop-Hintergrund durch das Symbol oder den Mauszeiger durchscheint. Das ist insbesondere bei Mauszeigern sinnvoll, bei denen normalerweise nicht der gesamte Bildbereich für den Mauszeiger selbst verwendet wird.
- Bei Benutzung der invertierten Farbe erscheint dieser Bereich in der »umgekehrten« Farbe wie der Hintergrund des Desktop während der Laufzeit.

 Wenn Sie zum Beispiel einen Mauszeiger in Form eines Fadenkreuzes in der invertierten Farbe erzeugen, wird er während der Laufzeit in invertierter Farbe zum Hintergrund des Desktop erscheinen. Positionieren Sie das Fadenkreuz über einen schwarzen Bereich des Desktop-Hintergrunds, so wird es weiß. Bewegen Sie es dann über einen roten Hintergrund, wird das Fadenkreuz blau dargestellt werden.

Die Benutzung der transparenten und invertierten Farbbereiche wird detaillierter in den Kapiteln 20 (»Symbole erzeugen«) und 21 (»Mauszeiger erzeugen«) beschrieben.

Transparente und invertierte Farben einstellen

Transparente und invertierte Farben erscheinen während der Laufzeit nicht in Ihrem Symbol oder Mauszeiger. Stattdessen werden Sie durch die Hintergrundfarbe bzw. ihre inverse Farbe ersetzt. Die Farben, die Sie als transparente und invertierte Farben verwenden, sollten diejenigen sein, die Sie nicht in Ihrem Mauszeiger oder Symbol verwenden wollen.

Die Standardeinstellung für die transparente Farbe ist die aktuelle Desktop-Hintergrundfarbe, wie Sie in der Systemsteuerung von Windows eingestellt ist.

Wird für Ihr Desktop eine gerasterte Farbe verwendet, so wählt Ressource Workshop die ähnlichste Farbe, die zur Verfügung steht. Arbeiten Sie mit einem Bildschirm mit 256 Farben, so trifft diese Einschränkung nicht zu, und die transparente und die Hintergrundfarbe des Desktops sind identisch.

Sie können die Voreinstellung ändern und jede beliebige Farbe als transparente Farbe definieren. Ihre Umdefinition ist jedoch nicht dauerhaft, und nach jedem Start von Ressource Workshop erscheint die Hinter-

grundfarbe als transparente Farbe. Nichtsdestotrotz bleiben mit transparenter Farbe eingefärbte Flächen transparent und werden in der aktuellen Farbe dargestellt, die als transparent gewählt wurde. Beachten Sie jedoch, daß durch die Farbänderung ggf. transparente und nichttransparente Bereiche Ihres Bildes in derselben Farbe angezeigt werden.

Um die Farben einzustellen, die der Paint-Editor als transparente und invertierte Farben verwendet, haben Sie folgende Möglichkeiten:

- Sie doppelklicken in der Farbpalette auf Transparent oder Invertiert.
- Sie wählen entweder die transparente oder die invertierte Farbe als Vorder- oder Hintergrundfarbe. Dann wählen Sie Symbol/Vordergrundfarbe bearbeiten oder Symbol/Hintergrundfarbe bearbeiten.

Ressource Workshop erzeugt das Dialogfenster Transparente Farbe bestimmen. Mit dem Dialogfenster wird ähnlich wie mit dem Farbe Bearbeiten Fenster umgegangen, nur daß die transparenten und invertierten Farben gleichzeitig gesetzt werden.

Das Fenster heißt immer Transparente Farbe bestimmen, unabhängig davon, ob Sie von der invertierten oder transparenten Farbe ausgegangen sind.

Beachten Sie die drei Farbfelder im oberen Bereich des Dialogfensters. Wenn Sie die RGB-Werte (Rot, Grün und Blau) verändern, wird die erzeugte Farbe als gewünschte Farbe dargestellt. Als zulässige Farbe erscheint die ähnlichste, im Fall von Darstellungen in 256 oder mehr Farben sogar dieselbe Farbe. Im invertierten Abschnitt wird die gerade bearbeitete Farbe invertiert dargestellt.

Die zulässige Farbe im Paint-Editor wird transparent und die Farbe im invertierten Abschnitt invertiert verwendet.

Farbpalette verbergen oder anzeigen

Um die Farbpalette zu verbergen, schließen Sie das Fenster wie ein gewöhnliches Windows-Fenster durch Doppelklicken ins Systemmenü-Symbol links in der Titelleiste der Farbpalette.

Sie können die Farbpalette durch Auswahl des Befehls Farbpalette verbergen schließen. Der Name des Menüs, in dem Sie den Befehl finden, hängt von der bearbeiteten Ressource ab. Erzeugen Sie ein Symbol, gehen Sie mit Symbol/Farbpalette verbergen vor. Erzeugen Sie einen Mauszeiger, müssen Sie mit Symbol/Farbpalette verbergen vorgehen.

Haben Sie die Farbpalette verborgen, erscheint statt des alten Befehls im Menü der Befehl Farbpalette anzeigen, und Sie können sich so die Farbpalette wieder darstellen lassen.

Farben selbst definieren

In Mauszeigern und Schriften können Sie keine Farben verändern.

Wenn Sie ein farbiges Bitmap oder ein Symbol bearbeiten, können Sie die Farbpalette verändern, um jede von Ihrem Bildschirmtreiber unterstützte Farbe aufzunehmen. Dies ist natürlich nur sinnvoll, wenn die Farbpalette alle darstellbaren Farben Ihres Computers enthält. Aber wenn Ihr Anzeigegerät 256 Farben unterstützt und Sie an einem Bild mit 16 Farben arbeiten, können Sie jede der 256 Farben in Ihre Farbpalette für die 16 Farben laden.

In den folgenden Abschnitten wird beschrieben, wie Sie eine beliebige Farbe aus der Farbpalette bearbeiten (einschließlich der transparenten und invertierten Farben).

Farben in der Farbpalette bearbeiten

Sie haben mehrere Möglichkeiten, eine Farbe auszuwählen:

- Sie doppelklicken in der Farbpalette auf die zu bearbeitende Farbe.
- Sie wählen entweder die zu bearbeitende Farbe als Vorder- oder Hintergrundfarbe. Dann wählen Sie Vordergrundfarbe bearbeiten oder Hintergrundfarbe bearbeiten aus dem Symbol- oder Bitmap-Menü.

Resource Workshop öffnet das Dialogfenster Farbe bearbeiten.

Palettenindex

Um Ihnen zu helfen, die Position einer Farbe in der Farbpalette zu bestimmen, enthält das Dialogfenster Farbe bearbeiten den Palettenindexwert. Jedes Kästchen in der Farbpalette ist mit einem Index numeriert, beginnend mit 0 in der linken oberen Ecke und dann zeilenweise von links nach rechts weiterzählend. Index-Werte werden nicht den transparenten und invertierten Farben zugeordnet.

In der 16-Farben-Palette zum Beispiel sind die Farben wie in folgender Abbildung durchnumeriert:

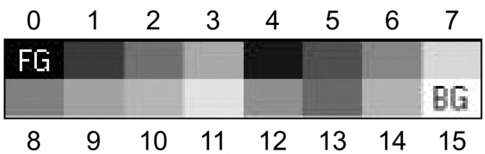


Abbildung 19.3: Der Farbenindex einer 16-Farben-Palette

Wenn Sie als Werkzeug Stift, Pinsel oder Airbrush benutzen, erscheinen auf der rechten Seite der Statuszeile der Palettenindex und die RGB-Werte für die Farbe unter dem Mauszeiger im Paint-Editor.

Eine Farbe bearbeiten

Um eine Farbe zu bearbeiten, können Sie deren RGB-Werte verändern, entweder indem Sie neue Werte statt des angezeigten in der linken Spalte eintragen oder indem Sie die Laufleisten auf der rechten Seite des Dialogfensters benutzen. Resource Workshop zeigt die ähnlichste darstellbare Farbe für die RGB-Werte im Gewünscht-Farbfeld.

Bei einer 256 (oder mehr)-Farben-Darstellung gibt es keinen Unterschied zwischen den Feldern Gewünscht und Vorhanden. Sie sehen immer die Farben selbst und nicht gerasterte Annäherungen.

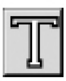
Bei einer 16-Farben-Darstellung könnte eine gerasterte Farbe in dem Feld auftauchen. Im Farbfeld zeigt Resource Workshop die ähnlichste Farbe, die bei Beachtung der Grenzen Ihres Anzeigegeräts noch darstellbar ist.

Bevor Sie den Paint-Editor verlassen, muß die Option Mit den Standardfarben des Geräts speichern unter Optionen/Editoroptionen ausgeschaltet sein, damit Ihre selbstdefinierte Farbpalette für zukünftige Verwendung erhalten bleibt. Sehen Sie unter Optionen/Editoroptionen nach, ob ein Markierungszeichen links neben Mit den Standardfarben des Gerätes speichern vorhanden ist.

Mit der Standardtaste wird die Farbe aus der Standard-Farbpalette (ein Windows-Objekt) geladen, die den gleichen Index wie die Farbpalette hat (wird am oberen Rand des Dialogfensters angezeigt).Mit der Systemtaste wird die Farbe aus der System-Farbpalette geladen, die den gleichen Index wie die Farbpalette hat (wird am oberen Rand des Dialogfensters angezeigt). Diese Taste ist für Standard-VGA-Geräte, die keine logischen Paletten unterstützen, ausgeschaltet. Sie ist verfügbar für Anzeigegeräte mit 256 oder mehr Farben.

Text in eine Bitmap-Zeichnung einfügen

Sie können Text in jede Bitmap-Zeichnung einfügen. Zum Beispiel können Sie ein Symbol mit Text versehen, um den Programmnamen zu repräsentieren.



Um Text einer Ressource hinzuzufügen, klicken Sie auf das Text-Werkzeug und anschließend in das Bild auf die Stelle, wo der Text beginnen soll. Hinter der blinkenden Eingabeaufforderung können Sie sodann Ihren Text eingeben.

Um die Standardausrichtung, die Schriften, die Größe und die Attribute (fett, kursiv oder unterstrichen) einzustellen, gehen Sie vor der Texteingabe mit Text/Linksbündig, Zentriert oder Rechtsbündig oder Text/Schrift vor. Um die Ausrichtung, Schriften, die Größe und die Attribute zu än-

dern, wählen Sie Text/Linksbündig, Zentriert oder Rechtsbündig oder Text/Schrift, bevor Sie wieder mit der Maus klicken.

Texteingaben oder Farbänderungen des Textes können Sie nur mit der linken Maustaste ausführen.

Zusätzlich können Sie nach der Texteingabe die Schriftfarbe ändern, wenn Sie vor einem erneuten Klicken mit der Maus eine neue Farbe aus der Farbpalette wählen.

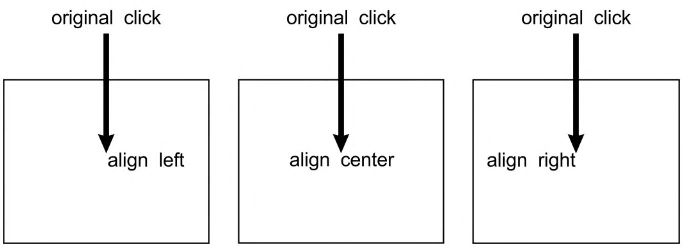
Sobald Sie auf ein anderes Werkzeug oder irgendwo in die Zeichnung klicken, können Sie am eingegebenen Text nichts mehr ändern. In dem Moment ist der Text Teil Ihres Bitmap geworden, so als ob Sie ihn dort gezeichnet hätten.

Text ausrichten

Um den Text auszurichten benutzen Sie Text/Linksbündig, Zentriert oder Rechtsbündig.

Diese Befehle steuern die Darstellung des Textes im Verhältnis zum Einfügepunkt (die Stelle, auf die Sie mit dem Text-Werkzeug geklickt haben). Sie können diesen Befehl vor oder unmittelbar nach der Texteingabe verwenden.

Abbildung 19.4: Text ausrichten.



Sie können die Ausrichtung des Textes nicht mehr ändern, nachdem Sie woanders geklickt haben, um etwas anderes auszuwählen.

Die Auswahl von Schrift, Größe und Textattributen

Wählen Sie Text/Schrift vor oder unmittelbar nach der Texteingabe, um die Darstellung des Textes zu ändern.

Sie können die Darstellung des Textes nicht mehr ändern, nachdem Sie woanders hingeklickt haben, um etwas anderes auszuwählen.

Nach Auswahl des Schrift-Befehls zeigt Ihnen Resource Workshop das Dialogfenster Schrift wählen.

Sie können den Schriftnamen einer Schrift, die Größe und Attribute Ihres Textes wählen. Sie könnten zum Beispiel Arial als Schrift, 12 Punkt als Größe und fett und kursiv als Attribute setzen. Die Buchstaben im Textfeld des Dialogfensters Schrift wählen ändern sich, um Ihnen zu zeigen, wie die Buchstaben Ihres Textes aussehen werden.

Die Pinselform bestimmen

Sie können mit dem Pinsel oder dem Airbrush in Ihrem Bitmap zeichnen. Dabei können Sie die Pinsel- oder Düsenform wählen.

Die aktuellen Pinsel- und Düsenformen werden immer am unteren Ende der Werkzeugpalette dargestellt (siehe weiter vorne in diesem Kapitel).

Sie haben folgende Möglichkeiten, eine andere Pinsel- oder Düsenform zu wählen:

- Benutzen Sie Optionen/Pinselform für die Pinselform.
- Benutzen Sie Optionen/Airbrush-Düse für die Düsenform
- Sie können auf die Darstellung der Pinsel- oder Düsenform in der Werkzeugpalette klicken.

Sie sehen ein Dialogfenster, welches Ihnen alle möglichen Pinselformen zur Auswahl anbietet. Oberhalb dieser Darstellung wird die aktuelle Form angezeigt. Klicken Sie auf eine der Formen und dann auf OK, um diese Pinsel- oder Düsenform auszuwählen.

Das nächste Mal, wenn Sie den Pinsel oder den Airbrush benutzen, wird die von Ihnen gewählte Airbrush-Düse oder Pinselform benutzt. Diese Form bleibt dieselbe, bis Sie wieder eine neue wählen.

Die Füllmuster bestimmen

Sie können ein Füllmuster bestimmen, mit dem folgende Werkzeuge in Ihrem Bild arbeiten:

- Pinsel
- Airbrush
- Ausgefüllter Rahmen
- Ausgefüllter abgerundeter Rahmen
- Ausgefüllte Ellipse

Die meisten Muster bestehen aus schwarzen Linien oder Punkten auf einem weißen Hintergrund. Wenn Sie mit der linken Maustaste zeichnen, werden die Punkte oder Linien statt in Schwarz durch die aktuelle Vordergrundfarbe und der weiße Hintergrund durch die aktuelle Hintergrundfarbe ersetzt. Beim Zeichnen mit der rechten Maustaste ist das Gegenteil der Fall: Die Punkte oder Linien werden statt in Schwarz in der aktuellen Hintergrundfarbe und der weiße Hintergrund in der aktuellen Hintergrundfarbe dargestellt.

Das aktuelle Muster wird in der rechten unteren Ecke der Werkzeugpalette angezeigt.

Sie können ein anderes Muster auf eine der folgenden Arten wählen:

- Mit Optionen/Muster
- Durch Klicken auf die Musteranzeige in der Werkzeugpalette

Sie sehen ein Dialogfenster, welches Ihnen alle möglichen Muster zur Auswahl anbietet. Oberhalb dieser Darstellung wird das aktuelle Muster angezeigt. Klicken Sie auf eines der Muster und dann auf OK, um diese Muster auszuwählen.

Das nächste Mal, wenn Sie ein Werkzeug benutzen, was mit Füllmustern arbeitet, wird das von Ihnen gewählte Muster benutzt. Dieses Muster bleibt dasselbe, bis Sie wieder ein neues wählen.

Die Strichstärke bestimmen

Sie können die Strichstärke bestimmen, mit der die folgenden Werkzeuge arbeiten:

- Stift
- Linie
- Leerer Rahmen
- Leerer abgerundeter Rahmen
- Leere Ellipse

Sie können die Strichstärke mit einer der beiden folgenden Methoden bestimmen:

- Mit Optionen/Stil des Stiftes
- Sie klicken auf die Strichstärkendarstellung in der Werkzeugpalette

Sie sehen ein Dialogfenster, das alle möglichen Strichstärken anzeigt, die Sie wählen können. Am oberen Rand des Dialogfensters zeigt Resource Workshop die Strichstärke an, die gerade ausgewählt wurde.

Beachten Sie dabei auch die Null-Strichstärkenwahl. Sie können Null wählen, wenn Sie ein ausgefülltes Rechteck (oder ein ausgefülltes abgerundetes Rechteck oder eine ausgefüllte Ellipse) ohne Randbereich erstellen wollen.

Die von Ihnen gewählte Strichstärke wird eingesetzt, wenn Sie das nächste Mal ein Werkzeug benutzen, das Linien zeichnet. Diese Strichstärke wird solange eingesetzt, bis Sie eine neue Strichstärke bestimmen.

Ausrichtung eines markierten Bereichs

Sie können einen markierten Bereich eines Bilds an den vier Rändern oder am Zentrum Ihres Editor-Fensters ausrichten. Die Ausrichtung des Bereichs bewegt ihn an die Stelle, die sie angegeben haben, so als ob Sie den Bereich markiert und dann mit der Maus verschoben hätten.Um einen bereits markierten Bereich der Zeichnung auszurichten, rufen Sie mit Optionen/Ausrichten das Dialogfenster Auswahl ausrichten auf.

Die Optionen unter Vertikal ermöglichen die Ausrichtung in der Senkrechten, die Optionen unter Horizontal ermöglichen die Ausrichtung in der Waagerechten.

Wenn Sie zum Beispiel Oben und Linke Seite auswählen und auf OK klicken, wird der markierte Bereich in die linke obere Ecke des Editor-Fensters bewegt. Wenn Sie unter Vertikal den Punkt Zentriert und Rechte Seite wählen, wird der Bereich zwischen dem oberen und unteren Rand des Fensters zentriert und dann soweit wie möglich nach rechts verschoben.

Der Paint-Editor verwendet sodann die aktuell gewählte Hintergrundfarbe zum Einfärben der Fläche, in der der Bereich ursprünglich vorhanden war (genauso, als ob Sie den Bereich mit der Maus verschoben hätten).

Skalieren eines markierten Bereichs

Sie können einen markierten Bereich Ihrer Zeichnung vergrößern, verkleinern oder verschieben. Dazu verwenden Sie das Werkzeug Markierungsrahmen setzen und die Befehle Optionen/Größe. Beim Skalieren des Bereichs dehnt oder staucht der Paint-Editor die Zeichnung innerhalb des markierten Bereichs.

Mit Optionen/Größe wird das Dialogfenster Auswahl dehnen aufgerufen.

Links werden unter Alte Position/Größe die aktuellen Pixelkoordinaten der linken oberen Ecke des markierten Bereichs sowie dessen Breite und Höhe in Pixeln angezeigt. Bei Erstaufruf des Fensters sind dieselben Werte auch unter Neue Position/Größe angegeben.

Um den Bereich zu skalieren und/oder zu verschieben (oder beides), geben Sie unter Neue Position/Größe neue Werte in die Felder ein. Mit der [*Tab*]-Taste können Sie von Feld zu Feld springen.

Um zum Beispiel einen rechteckigen Bereich in die linke obere Ecke zu bewegen und ihn 30 Pixel breit und 5 Pixel hoch zu machen, müssen Sie folgendes unternehmen:

1. Mit dem Werkzeug Markierungsrahmen setzen markieren Sie den Bereich, den Sie bewegen und verschieben wollen.
2. Geben Sie 0 neben Links ein, und drücken Sie [*Tab*].
3. Geben Sie 0 neben Oben ein, und drücken Sie [*Tab*].
4. Geben Sie 30 neben Breite ein, und drücken Sie [*Tab*].
5. Geben Sie 5 neben Höhe ein.
6. Drücken Sie die [*Eingabe*]-Taste, oder klicken Sie auf OK.

Der markierte Bereich ist nun in der linken, oberen Ecke des Rahmens mit neuer Höhe und Breite positioniert. Ursprünglich vom Bereich belegte Flächen werden mit der Hintergrundfarbe eingefärbt.

Allgemeine Optionen des Paint-Editors einstellen

Um die allgemeinen Optionen des Paint-Editors einzustellen, rufen Sie mit Optionen/Editoroptionen das Dialogfenster Optionen für Paint-Editor setzen auf

Aktualisierung beider Bilder

Haben Sie zwei Ansichten derselben Zeichnung im Paint-Editor, können Sie wählen, wann das Bild im nicht aktiven Abschnitt aktualisiert wird. Ist die Option Aktualisierung beider Bilder gewählt, wird das zweite Bild sofort auf den neuesten Stand gebracht, während Sie zeichnen.

Ist die Option nicht markiert, wird das zweite Bild erst dann auf den neuesten Stand gebracht, wenn Sie mit dem Zeichnen fertig sind. Wenn Sie dann zum Beispiel mit einem Pinsel über das Bild ziehen, zeigt der Paint-Editor das Ergebnis zunächst nur im aktiven Fenster an. Sobald Sie aber die Maustaste loslassen, wird der Strich auch im zweiten Bild gezeichnet.

Auf vergrößerten Fenstern können Sie sich ein Gitter anzeigen lassen, an dem zu sehen ist, wie das Bild auf der Pixelebene gezeichnet ist.

Mit den Standardfarben des Gerätes speichern

Schalten Sie diese Option aus, wenn Sie eine selbstdefinierte Farbpalette speichern wollen. Ist die Option markiert, wird jede veränderte Farbe in der Farbpalette beim Verlassen des Paint-Editors auf die Standardfarbe zurückgesetzt.

Sie können selbstdefinierte Farben nur speichern, wenn Ihr Anzeigegerät 256 Farben darstellen kann und logische Paletten unterstützt. Weiter vorne in diesem Kapitel finden Sie eine Beschreibung, wie eigene Farbpaletten zu definieren sind.

K A P I T E L 20

Symbole erzeugen

Symbole sind kleine Bitmap-Graphiken in der Größe von 64x64-, 32x32- oder 32x16-Pixeln.

Windows-Programme benutzen Symbole, um verkleinerte Fenster darzustellen.

Mit dem Paint Editor können Sie Symbole gestalten.

Der Paint Editor umfaßt verschiedene Hilfsmittel zum Zeichnen und eine leicht zu benutzende Farbenpalette zur Farbgestaltung Ihrer Symbole. Sie können Ihre Graphik vergrößern und sich mehrere Ansichten des Symbols zeigen lassen. In Kapitel 19 wird die Benutzung des Paint Editor erläutert.

Die Arbeit mit Symbolen erfolgt in vier grundlegenden Schritten.

1. Aufruf des Paint-Editor.
 Wenn Sie eine neue Symbol-Ressource erzeugen, erzeugt der Paint Editor für Sie ein leeres Gitter zum Bearbeiten. Wenn Sie ein bestehendes Symbol bearbeiten, erscheint es im Paint Editor.
2. Erzeugen oder Bearbeiten des Symbols.
3. Symbol testen.
4. Symbol speichern.

Neue Symbole erzeugen

Sie können neue Symbole einer Projektdatei zuordnen oder eine neue Symbol-Datei erstellen, die die Dateiendung .ICO erhält.>ihrer Projekt Datei zufügen, oder Sie können eine eigenständige Symboldatei mit der Erweiterung .ICO erzeugen.

Hinzufügen eines Symbols zur Projekt Datei

Um ein Symbol einer .RC-(oder .DLG-) Datei hinzuzufügen:

1. Öffnen Sie ein neues oder bereits bestehendes Projekt. (Kapitel 2 beschreibt, wie Sie ein Projekt öffnen).
2. Mit *Ressource/Neu* wird das Dialogfenster *Neue Ressource* angezeigt.
3. Durchsuchen Sie die Liste *Ressourcen-Typ* und markieren Sie ICON. Der Name der aktuellen Projekt-Datei erscheint im Fenster unter *Ressource einfügen in*. Sie können aus der Liste auch eine andere Projekt-Datei wählen (sofern eine vorhanden ist), oder Sie klicken auf OK, um die bestehende Auswahl zu bestätigen.
4. Danach müssen Sie entscheiden, ob Sie das Symbol in Quelltext oder als Microsoft-kompatible Binärdatei erstellen wollen.
 Wählen Sie *Quelle*, so erzeugen Sie ein Symbol, daß in eine Projekt-Datei eingebunden wird. Das Dialogfenster »*Neues Symbol*« erscheint.
 Wählen Sie Binärformat, so wird eine eigenständige .ICO-Datei erstellt.
5. Im Dialogfenster "*Neues Symbol*" können Sie die Größe des neuen Symbols in Pixeln sowie die Anzahl der Farben angeben, die unterstützt werden sollen. Befindet sich eine Bitmap-Datei in der Zwischenablage, werden deren Größe und Farben als Voreinstellung übernommen

Wählen Sie die Bildgröße und die Farbanzahl und klicken Sie auf OK. Resource Workshop fügt den Namen des neuen Symbols in die Projekt-Datei ein und ruft den Paint Editor auf.

Erzeugen einer seperaten Symbol-Datei

Sie können eine separate *Symbol-Datei* auf eine der folgenden Arten erzeugen:

- Wählen Sie im Dialogfenster bei der Frage nach Binärdatei oder Quelltext die Option *Binärdatei*.
- Wählen Sie "*Datei/Neues Projekt*" und anschließend .ICO aus dem Dialogfenster.

Um eine eigenständige Symbol-Datei in eine Projekt-Datei einzubinden, öffnen Sie die Projekt-Datei und wählen "*Datei/Dem Projekt hinzufügen*".

Option Binärdatei

Entscheiden Sie sich für das Erstellen einer Binärdatei, zeigt Resource Workshop das Dialogfenster "*Neue Ressourcendatei*" an. Im Fenster müßen Sie den Namen ihres Symbols sowie den Namen der Datei angeben, in die ein Verweis auf das Symbol eingefügt wird. Wenn der gewählte Ressourcentyp ein Symbol ist, wird im Feld "Dateiname" *.ICO angezeigt.

1. Geben Sie ins Feld "*Dateiname*" einen Namen für Ihre .ICO-Datei ein. Im Feld "Ein Verweis zu der Ressource..." muß der Name Ihres Projekts

angegeben werden. Um das Verzeichnis zu wechseln, in dem Ihre Datei gespeichert wird, doppelklicken Sie auf die entsprechenden Symbole im Feld "*Verzeichnisse*". Der gewählte Pfad wird in dem Feld "Pfad" angezeigt.

2. Haben Sie alles in den gewünschten Zustand gebracht, klicken Sie auf OK. Ein Dialogfenster erscheint mit dem Hinweis, daß die Datei noch nicht existiert und fragt Sie, ob die Datei jetzt erzeugt werden soll. Klicken Sie auf "Ja".
3. Das Fenster "*Neue Ressource*" wird verlassen und das Fenster "Neues Symbol" erscheint. Wählen Sie "*Größe*" und "*Farbenzahl*", klicken auf OK und der Paint Editor wird aufgerufen.

 Sie können auch den Namen einer bestehenden Datei aus dem Dateifenster übernehmen. In dem Fall wird aber der Inhalt dieser Datei überschrieben. Vorsichtshalber wird daher nachgefragt, ob die Datei wirklich überschrieben werden soll.

Symbol Projektdatei

Mit "*Datei/Neues Projekt*" und der Auswahl von .ICO im Dialogfenster "*Neues Projekt*" können Sie automatisch eine eigenständige Symboldatei erzeugen. In dem Fall wird sofort das Dialogfenster "*Neues Symbol*" angezeigt. Nach Angabe von Farben und Größe wird der Paint Editor aufgerufen.

Symbole bearbeiten

Im folgenden Abschnitt werden verschiedene Möglichkeiten zum Bearbeiten Ihres Symbols aufgezeigt.

Ansicht verschiedener Auflösungen

Die Standard-EGA/VGA-Auflösung ist 32x32, aber Sie können Symbole auch in zwei anderen Auflösungen erzeugen: 32x16 und 64x64. Im Menü "*Ansicht*" sind Anweisungen verfügbar, mit denen Sie sich das Symbol in anderen Auflösungen ansehen können.

- Erzeugen Sie ein Symbol in 32x32-Auflösung, können Sie mit "Ansicht/CGA-Auflösung" [32x16] das Symbol betrachten, wie es auf einem CGA-Bildschirm zu sehen sein wird.
- Erzeugen Sie ein Symbol in 32x16-CGA-Auflösung, können Sie mit Ansicht/EGA/VGA-Auflösung [32x32] das Symbol betrachten, wie es auf einem EGA- oder VGA-Bildschirm zu sehen sein wird.
- Erzeugen Sie ein Symbol in 64x64-Auflösung, sind beide Ansichtsmöglichkeiten gegeben. Sie können das Symbol betrachten, wie es auf einem CGA-, EGA- oder VGA-Bildschirm zu sehen wäre.

Mit diesen Anweisungen ändern Sie nur die Ansichtsweisen des Symbols, nicht das Symbol selbst. Das Symbol liegt nach wie vor in derselben Auflösung vor.

Benutzen transparenter oder invertierter Farbbereiche

Sie können transparente und invertierte Farbbereiche in Ihren Symbolen verwenden. Während der Laufzeit werden transparente Farbbereiche durchsichtig, so daß durch sie hindurch der Hintergrund sichtbar wird. Farbbereiche des Symbols, die mit invertierter Farbe erstellt sind, haben zum Hintergrund die entgegengesetzte Farbe - z.B. weiß über schwarz oder grün über rot.

Invertierte Farben sind beim Erzeugen von Mauszeigern besonders nützlich.

Wenn Sie einen "Mauszeiger" erzeugen, sehen Sie zunächst ein Gitter angefüllt mit transparenter Farbe . Wenn Sie eine Linie mit einer anderen Farbe ziehen und den Mauszeiger testen, sehen Sie nur diese Linie vor dem Hintergrund. Sie sehen weder das Gitter noch den Bereich, der mit transparenter Farbe angefüllt ist.

Hinzufügen eines Bildes zu einer Symbolressource

Im allgemeinen werden Sie eine neue Ressource für jeden Symbolentwurf erzeugen und nicht verschiedene Symbole in dieselbe Ressource einfügen. Aber wahrscheinlich wollen Sie verschiedene Farbentwürfe desselben Symbols in einer Ressource speichern. Die Farbvariationen eines Symbols werden *Bilder* genannt. Wollen Sie z.B. eine 2-farbige und eine 16-farbige Version desselben Entwurfs speichern, können Sie diese Bilder in derselben Symbolressource speichern.

Der Grund, daß eine Symbolressource verschiedene Farbvariationen desselben Symbols unterstützt, ist, daß Windows diejenigen Farben auswählt, die am besten durch die vorhandene Hardware unterstützt werden. Windows wählt 2-farbiges Anzeigen für Monochrom-Monitore und 16-farbiges Anzeigen bei VGA-Darstellung.

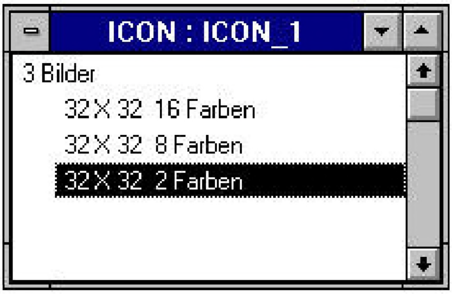
Windows 3.x unterstützt nicht die 256 Farbendarstellung eines Symbols, selbst wenn es Ihre Hardware erlaubt. Ihr Programm muß die Unterstützung für 256 Farben bereitstellen.

Um ein Bild einer bestehenden Ressource hinzuzufügen:

1. Öffnen Sie ein Projekt.
2. Doppelklicken Sie auf den Eintrag des Symbols im Projektfenster, oder markieren Sie ihn und wählen Sie "Ressource/Bearbeiten". Das Symbolfenster erscheint.

3. Um eine neue Version eines bestehenden Symbols zu erstellen, wählen Sie "Bilder/Neues Bild". Das Dialogfenster "Neues Symbol" erscheint, und Sie können die Bildgröße und Farben wählen.

Abbildung 20.1: Dialogfenster Neues Symbol



4. Wählen Sie dieselbe Größe, aber eine andere Farbenanzahl als das ursprüngliche Symbol. Klicken Sie auf OK. Der neue Symboleintrag erscheint im Projektfenster.
5. Doppelklicken Sie auf den Eintrag, oder markieren Sie ihn und wählen Sie "Bilder/Bild bearbeiten".
6. Normalerweise öffnen Sie als nächstes ein bestehendes Symbolbild und kopieren es in das (immer noch leere) Bild. Möglicherweise müssen Sie danach das Bild bearbeiten, wenn die Farben in einer Weise umgesetzt werden, daß sich die Erscheinungsform des Symbols verändert.

Eigenschaften eines Symbols verändern.

Um die Eigenschaften - Auflösung oder Farbanzahl - des aktuellen Symbols zu ändern:

1. Wählen Sie "*Symbol/Größe und Attribute*" und das Fenster "*Symbol-Attribute*" erscheint.

 Abgesehen von einer zusätzlichen Taste, die im weiteren beschrieben wird, ist das Fenster "*Symbol-Attribute*" identisch mit dem Dialogfenster "*Neues Symbol*".

2. Wählen Sie eine Auflösung oder Farbpalette und klicken Sie auf OK.

➥ Das Verändern der Symbolattribute verändert tatsächlich das Symbol selbst, während das Symbol bei Ansehen in einer anderen Auflösung unverändert ist.

Anzeigen von Geräteinformationen

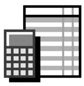
Die zusätzliche Taste im Fenster "*Symbol-Attribute*" ist mit *Info* beschriftet und dient der Anzeige von Geräteinformationen. Beim Klicken auf diese Taste erscheint das Fenster "*Geräteinformationen anzeigen*", welches folgende Informationen über Ihre Bildschirmkonfiguration aufweist:

- Anzahl der Bits pro Pixel
- Anzahl der Farbebenen
- Anzahl der anzeigbaren Farben
- Ob das Gerät mit Farbpaletten arbeitet

Wenn Ihre Anzeigekonfiguration logische Farbpaletten unterstützt, werden im Fenster "*Geräteinformationen*" zusätzlich folgende Informationen angezeigt:

- Größe der Farbpalette
- Anzahl der reservierten Farben in der Systempalette
- Die aktuelle Farbauflösung der Bildschirmkonfiguration in Bits pro Pixel

Erzeugen eines Beispielsymbols



Dieser Abschnitt beschreibt, wie das Beispielsymbol, das links dargestellt ist, erzeugt wird.

Um ein neues Symbol zu erzeugen, müssen Sie zuerst ein Projekt öffnen und dann folgende Schritte ausführen, um den Paint Editor mit einem neuen, leeren Bild aufzurufen

Kapitel 14 beschreibt, wie ein Projekt geöffnet wird.

1. Wählen Sie "*Datei/Neues Projekt"*, um ein neues Projekt zu erzeugen oder, öffnen Sie ein bestehendes Projekt.
2. Mit "*Ressource/Neu*" weisen Sie Resource Workshop an, eine neue Ressource anzulegen. Bei der Frage nach dem Ressourcentyp wählen Sie ICON.
3. Wählen Sie *Quelle*, um das Symbol als Ressourcenscript in einer .RC-Datei zu speichern.
4. Das Dialogfenster "*Neues Symbol*" erscheint. Wählen Sie 32x32 als Größe und 16 Farben und klicken Sie dann auf OK.
5. Der Paint Editor wird aufgerufen.

Ein Taschenrechnersymbol zeichnen

Um das Taschenrechnersymbol zu zeichnen,

1. Wählen Sie "*Optionen/Editor Optionen*" und markieren Sie Gitter auf vergrößerte Fenster, damit Sie es leichter bei der Ausrichtung der Taschenrechnertasten haben, und klicken Sie auf OK.

Das Gitter zeigt Ihnen die einzelnen Pixel, wenn Sie einen Bildausschnitt des Symbols im Paint Editor vergrößern. Sie sehen leichter, was Sie tun, wenn Sie mit vergrößerten Ausschnitten des Symbols arbeiten.

2. Das Symbol bleibt in der rechten Hälfte des Fensters in der Originalgröße. Um das Bild in der linken Hälfte zu vergrößern, doppelklicken Sie auf das Vergrößern-Symbol in der Werkzeugpalette, bis das Bild so groß geworden ist, daß es gerade noch vollständig auf dem Bildschirm sichtbar ist.

 Jedes Gitterquadrat der vergrößerten Ansicht repräsentiert ein einzelnes Pixel.

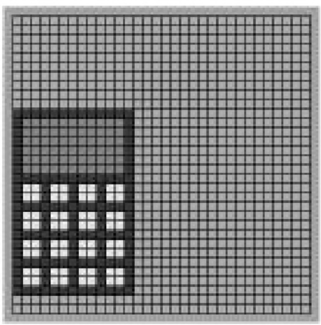
 Nun können Sie mit dem Zeichnen beginnen.

3. Wählen Sie als Farbe für den Taschenrechner Dunkelrot. Wenn Sie auf Dunkelrot in die Farbenpalette klicken, sehen Sie die Buchstaben *VG* (Vordergrund) auf Dunkelrot.
4. Klicken Sie auf das schraffierte Rechteck in der Werkzeugpalette. Es befindet sich genau unter dem Texterstellungswerkzeug, welches ein großes T als Symbol hat.
5. Zeichnen Sie den Taschenrechner links unten im Symbol.

Lassen Sie unter dem Taschenrechner zwei Pixelreihen für die Schattierung frei.

Sie können das gefüllte Rechteck benutzen, um alle Bestandteile des Taschenrechners zu zeichnen - das Gehäuse, die Anzeige und die Tasten. Wählen Sie verschiedene Farben für Anzeige und Tasten, indem Sie in die Farbenpalette klicken, bevor Sie zeichnen (Versuchen Sie mal Gelb für die Tasten und Dunkelgrün für die Anzeige). Wenn Sie fertig sind, sollte Ihr Taschenrechner ähnlich wie der folgende aussehen:

Abbildung 20.2: Vergrößertes Bild des Taschenrechners vor Schattierung



Haben Sie einen Fehler gemacht, können Sie mit "*Bearbeiten/Rückgängig Zeichnen*" oder [*Alt+BckSp*] korrigieren. Sie können auch das "*Pfeilwerkzeug*" zum Markieren eines rechteckigen Bildschirmausschnitts oder die Schere benutzen, um einen Ausschnitt des Bildes zu definieren und zu löschen

(mit der [*Del*] Taste oder "*Bearbeiten/Löschen*"). Sie können auch den Radiergummi aus der Werkzeugpalette benutzen.

Wenn Sie alles löschen und von vorne beginnen wollen, doppelklicken Sie einfach auf das Werkzeug Radiergummi.

Dreidimensionale Effekte erzeugen

Damit der Taschenrechner dreidimensional aussieht:

1. Wählen Sie Schwarz als Farbe des Schattens.
2. Benutzen Sie das Liniewerkzeug, um eine Schattenlinie entlang der rechten Seite und am unteren Ende des Taschenrechners einzufügen. Die Schattenlinie soll zwei Pixel breit sein und zwei Pixel unter dem oberen Rand sowie zwei Pixel vor dem linken Rand des Taschenrechners enden.
3. Sie können auch ein dreidimensionales Aussehen für das Fenster erreichen, wenn Sie eine weiße Linie den linken und oberen Rand des Fensters entlang zeichnen.

So sollte der Taschennrechner nun aussehen:

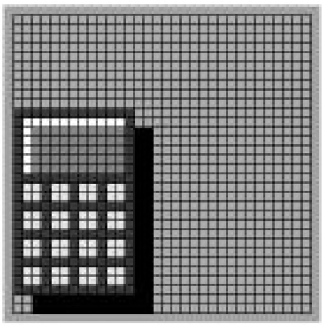


Abbildung 20.3: Vergrößertes Bild des schattierten Taschenrechners

Wenn Sie mit dem Zeichnen fertig sind, können Sie mit "Datei/Projekt speichern" Ihr neues Symbol speichern lassen. Es ist eine gute Angewohnheit, frühzeitig und häufig seine Arbeit zu speichern.

Zeichnen der Haushaltsbuchseite

1. Wählen Sie Schwarz und das Linienwerkzeug.
2. Zeichnen Sie mit dem Linienwerkzeug eine senkrechte schwarze Linie, die 5 Pixel rechts links vom rechten, oberen Rand des Taschenrechners beginnt und zum oberen Ende des Fensters führt, dann weiter zum äußersten rechten Rand gezogen wird und schließlich nach unten zwei Pixel vor den unteren Rand des Fenster führt. Von dort wird sie nach links gezogen, bis sie den Schatten des Taschenrechners berührt. Sie haben soeben den Umriß der Haushaltsbuchseite erstellt.

3. Wählen Sie die weiße Farbe und das Farbeimerwerkzeug. Klicken Sie den Farbeimer auf der Haushaltsbuchseite an, um die Seite mit weißer Farbe zu füllen.

4. Wählen Sie Hellblau und das Linienwerkzeug.

5. Klicken Sie auf Strichstärke des Stiftes im rechten unteren Eck der Werkzeugpalette und wählen Sie im Dialogfenster "*Aktuelle Strichstärke*" die Zwei Pixel-Breite (rechts von null).

Mit "*Bearbeiten/Rückgängig Zeichnen*" können Sie alle Fehler berichtigen.

6. Sie zeichnen zum Beginn zwei Pixel vom unteren Rand der Haushaltsbuchseite und einen Pixel von dort, wo die Linie losgehen soll (da sie zwei Pixel breit ist) eine Serie von zwei Pixel hohen Linien über die gesamte Breite des sichtbaren Teils der Haushaltsbuchseite. Jede Linie sollte zwei Pixel oberhalb der unteren Linie dargestellt werden. (Die unteren drei Linien werden links vom Taschenrechnerschatten begrenzt.

7. Wählen Sie Hellrot und das Linienwerkzeug.

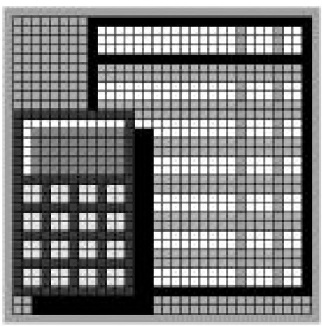
8. Klicken Sie auf Strichstärke des Stiftes im rechten unteren Eck der Werkzeugpalette und wählen Sie im Dialogfenster "*Aktuelle Strichstärke*" die Ein Pixel-Breite (direkt oberhalb von null).

9. Drei Pixel vom rechten Rand des Haushaltbuches zeichnen Sie eine vertikale Linie die ganze Länge der Seite hinauf. Wiederholen Sie dies vier Pixel zur linken der Linie.

10. Wählen Sie Schwarz und zeichnen Sie eine horizontale Linie, die unmittelbar über der obersten hellblauen Fläche beginnt und die Weite der Seite entlangführt.

11. Wenn Sie mit Ihrem Symbol fertig sind, sollte das Paint-Editor-Fenster wie folgt aussehen:

Abbildung 20.4: Symbol der Haushaltsbuchführung



KAPITEL 21

Mauszeiger erzeugen

Mauszeiger sind 32x32 Bitmap-Bilder, die die Position der Maus auf dem Bildschirm repräsentieren. Eine Windows-Anwendung verwendet gewöhnlich mehrere Mauszeigerformen, um verschiedene Funktionen des Programms darzustellen.

Windows bietet Ihnen eine Auswahl von Standardmauszeigern, die Sie in Ihrer Anwendung verwenden können. Zusätzlich können Sie eigene Mauszeiger erzeugen, die bestimmte Funktionen Ihres Programms darstellen. Ein solcher Mauszeiger wäre zum Beispiel der Farbeimer, der angezeigt wird, wenn Sie das Farbeimer-Werkzeug im Paint-Editor verwenden.

Um persönliche Mauszeiger zu gestalten, wird der Paint-Editor benutzt (siehe Kapitel 19).

Die Arbeit mit Mauszeigern erfolgt in vier grundlegenden Schritten:

1. Den Paint-Editor aufrufen.

 Wenn Sie eine neue Mauszeiger-Ressource erzeugen, zeigt Ihnen der Paint-Editor ein leeres Gitter, mit dem Sie arbeiten können. Wollen Sie einen bestehenden Mauszeiger verändern, erscheint dieser im Paint-Editor.

2. Den Mauszeiger erzeugen oder bearbeiten.
3. Den Mauszeiger testen.
4. Den Mauszeiger speichern

Einen neuen Mauszeiger erzeugen

Sie können einen neuen Mauszeiger an Ihre Projektdatei anfügen, oder Sie können eine eigenständige Mauszeigerdatei mit der Dateierweiterung .CUR erzeugen.

Einen Mauszeiger der Projektdatei hinzufügen

Um einen Mauszeiger einer .RC-(oder .DLG-)Datei hinzuzufügen,

1. öffnen Sie ein neues oder bestehendes Projekt.
2. Wählen Sie *Ressource/Neu*. Das Dialogfenster *Neue Ressource* wird angezeigt.
3. Markieren Sie CURSOR als Ressourcentyp.
 Der Name der aktuellen Projektdatei erscheint unter »*Ressource einfügen in*«. Sie können dieses Listenfenster durchsuchen und sich eine andere Datei aussuchen (falls eine andere aufgelistet ist), oder klicken Sie auf *OK*, um die Auswahl der angezeigten Datei zu bestätigen.

Der Unterschied zwischen Ressourcen, die direkt eingefügt werden, und denen, auf die nur verwiesen wird, wird im entsprechenden Abschnitt ausführlich dargestellt.

4. Resource Workshop fragt, ob Sie den Mauszeiger in Quellenform oder in Microsoft kompatiblem Binärformat erzeugen wollen.
5. Wählen Sie *Quelle*, um einen Mauszeiger zu erzeugen, der in die Pro--jektdatei eingefügt wird. Der Paint-Editor wird aufgerufen.

Wenn Sie *Binär* wählen, wird eine eigenständige .CUR-Datei erzeugt.

Eine eigenständige Mauszeigerdatei erzeugen

Sie können eine eigenständige Mauszeigerdatei auf folgende Weisen erzeugen:

- Sie wählen *Binär* im Dialogfenster, welches fragt, ob Sie die Datei als Quelltext oder Binärdatei erzeugen wollen.
- Sie markieren unter *Datei/Neues Projekt* im Dialogfenster die Auswahl .CUR *Neues Projekt*.

Um einen Verweis auf eine selbstständige Mauszeigerdatei in Ihre Projektdatei einzufügen, öffnen Sie Ihr Projekt und wählen dann *Datei/Dem Projekt hinzufügen*.

Datei im Binärformat

Wenn Sie *Binärformat* auswählen, wird Ihnen das Dialogfenster *Neue Ressourcendatei* angezeigt. Hier können Sie Ihre Mauszeigerdatei benennen und die Projektdatei angeben, die einen Verweis auf Ihren Mauszeiger enthalten wird. Da Sie einen Mauszeiger erzeugen, zeigt das Dateiname-Eingabefeld *.CUR an.

1. Geben Sie einen Dateinamen mit Erweiterung .CUR in das Eingabefeld für den Dateinamen ein. Der Name Ihres Projekts sollte unter »*Ein Verweis zu der Ressource in dieser Datei wird eingefügt in*« stehen. Um das Verzeichnis zu wechseln, in dem Ihre Datei gespeichert wird, geben Sie einen neuen Pfad durch Doppelklicken auf die entsprechenden Symbole im Listenfeld Verzeichnisse an. Im Textfeld *Pfad* sehen Sie den Pfad, den Sie auf diese Weise setzen.

2. Sind Sie mit allem fertig, so klicken Sie auf OK. Ein Dialogfenster informiert Sie, daß die von Ihnen angegebene Datei noch nicht existiert, und fragt, ob Sie diese Datei jetzt erzeugen wollen. Hier klicken Sie auf Ja.
3. Wenn Sie das Dialogfenster *Neue Ressource* verlassen, wird der Paint-Editor aufgerufen.

Sie könnten auch eine Datei aus der Dateiliste auswählen. Da Sie in dem Fall aber eine bestehende Datei überschreiben, wird vorsichtshalber Ihre Zustimmung zu dieser Aktion erfragt.

Mauszeiger-Projektdatei

Wenn Sie *Datei/Neues Projekt* wählen und .CUR im Fenster *Neues Projekt* markieren, wird automatisch eine eigenständige Datei erzeugt. Der Paint-Editor wird sofort aufgerufen.

Mauszeiger bearbeiten

Die folgenden Abschnitte beschreiben mehrere Methoden, wie Sie Ihre Mauszeiger bearbeiten können.

Farbpalette für Mauszeiger

Die Farbpalette für Mauszeiger unterscheidet sich in einem Aspekt grundlegend von den Farbpaletten der anderen Bitmap-Ressourcen: Es gibt nur zwei Vorder- und Hintergrundfarben, nämlich Schwarz und Weiß. Ein typischer Mauszeiger ist dann auch eine weiße Form mit schwarzer Umrandung, wie z.B. ein Pfeil oder ein Fragezeichen.

Mit invertierten und durchsichtigen Farben arbeiten

Sie können durchsichtige und invertierte Farbflächen in Ihrem Mauszeiger verwenden. Während der Laufzeit erscheinen durchsichtige Flächen insofern unsichtbar, als an ihrer Stelle der Hintergrund des Desktop dargestellt wird. Mit invertierter Farbe gemalte Flächen nehmen die gegensätzliche Farbe zum Desktophintergrund an, z.B. Weiß über Schwarz. Für den Kontaktpunkt (»Hot Spot«) des Mauszeigers werden oft invertierte Farben benutzt.

Zu Beginn ist Ihr neuer Mauszeiger nur ein durchsichtig eingefärbtes Gitter. Wenn Sie eine Linie in einer anderen Farbe zeichnen und dann den Mauszeiger testen, sehen Sie als Ergebnis nur die Linie vor dem Desktop-Hintergurund. Sie sehen also weder das Gitter noch die durchsichtigen Bereiche.

Es ist nicht empfehlenswert, daß Sie Ihre durchsichtigen oder invertierten Farben auf Schwarz oder Weiß setzen: Ihr Cursor wechselt dann entwe-

der ständig die Farbe, wenn der Anwender ihn über den Bildschirm bewegt, oder er wird gänzlich unsichtbar.

Den Kontaktpunkt des Mauszeigers setzen

Eine wichtige Entscheidung bei der Bearbeitung Ihres Mauszeigers ist die Frage, wo der Kontaktpunkt in der Grafik angesiedelt werden soll. Der Kontaktpunkt ist der Punkt Ihrer Grafik, welcher die Stellung des Mauszeigers auf dem Bildschirm bestimmt, wenn der Anwender den Mauszeiger positioniert und klickt, um eine Auswahl zu treffen.

Um einen Kontaktpunkt festzulegen, müssen Sie sich über die genauen Pixelkoordinaten des Kontaktpunkts klarwerden . Pixelkoordinaten werden in waagrechten (x) und senkrechten Einheiten (y) gemessen. Der oberste Pixel am linken Rand hat die Koordinaten x=0 und y=0. Der unterste Pixel am rechten Rand hat für einen 32x32 Pixel-Mauszeiger die Koordinaten x=31 und y=31, und für einen 32x16-Mauszeiger die Koordinaten x=31 und y=15.

Um den Kontaktpunkt festzulegen, gehen Sie folgendermaßen vor:

1. Sie vergrößern den Mauszeiger im Paint-Editor solange, bis Sie bequem die Pixelkoordinaten für den Kontaktpunkt genau festlegen können.
2. Ein Gitter sollte auf dem vergrößerten Bild sichtbar sein. Ist dies nicht der Fall, markieren Sie unter *Optionen/Editoroptionen* die Auswahl *Gitter auf vergrößerte Fenster*.
3. Wählen Sie ein Zeichenwerkzeug, mit dem Sie pixelgenau zeichnen können. Das Werkzeug Linie ist eine gute Wahl, da Sie mit dem Fadenkreuz genau auf den gewünschten Punkt fahren können.
4. Zeigen Sie mit dem Werkzeug auf den Punkt in Ihrer vergrößerten Zeichnung, an dem Sie den Kontaktpunkt definieren wollen. Notieren Sie sich die Koordinaten, die dabei in der Statuszeile angezeigt werden.
5. Unter *Mauszeiger/Kontaktpunkt* setzen geben Sie diese Koordinaten ein.

K A P I T E L 22

Erstellen von Schriften

In diesem Handbuch werden »Schrift« und »Schrift-Ressource« gleichwertig benutzt.

Üblicherweise bezeichnet man mit Schrift eine Menge von Zeichen bestimmter Größe und Stils. Eine Schrift-Ressource ist eine Sammlung von Daten, die es einem Computer ermöglicht, einzelne punktweise vorliegende Bilder (Buchstaben oder Ähnliches) auf Geräten wie Bildschirm oder Drucker auszugeben.

Die Schrift enthält Daten, die für die gesamte Menge gelten, wie Name, Standardgröße, Zeichensatz usw. In ihr befinden sich auch die Informationen, die der Rechner zur Ausgabe eines einzelnen Bildes benötigt.

Die Arbeit mit Schrift-Ressourcen besteht aus fünf Hauptschritten:

1. Starten des Bitmap-Editors mit einer neuen oder einer bereits existierenden Schrift.
2. Erstellen oder Bearbeiten des Schriftbildes mit dem Bitmap-Editor.
3. Speichern der Schrift-Ressource als Teil einer Projektdatei oder als eigene Datei.
4. Beenden des Resource Workshop. Hinzufügen der Schrift-Ressource in eine spezielle DLL-Datei mit der Erweiterung .FON, die nur Ressourcen aufnimmt.
5. Einfügen eines Aufrufs dieser .FON-Datei im Programm, kompilieren und Testen der Schrift-Ressource.

Schrifttypen

Windows unterstützt zwei Grundtypen von Schriften: Raster- und Outline-Schriften. Resource Workshop arbeitet ausschließlich mit Rasterschriften.

- Rasterschriften bestehen aus einer Bitmap für jedes Zeichen.

- Outline-Schriften enthalten für jedes Zeichen eine Reihe von Zeichenbefehlen. Normalerweise enthalten sie auch sogenannte »Hinweise«, Informationen, dies dem Computer ermöglichen, Schriften auf verschiedenen Geräten und in unterschiedlichen Größen ohne merklichen Qualitätsverlust auszugeben. Bei den Outline-Schriften von Windows fehlen diese Hinweise. Outline-Schriften mit diesen Einschränkungen werden Vektorschriften genannt; Beispiele dafür sind die unter Windows verfügbaren Schriften Roman, Script und Modern. Als Windows-Anwender steht Ihnen durch Drittanbieter aber auch die fortschrittlichere Technologie der Schriftrasterung zur Verfügung.

Warum werden Bilder in Schriften abgelegt?

Obwohl Sie mit Windows-Resource Workshop benutzerdefinierte Zeichen erstellen können, werden Sie für solche Aufgaben wahrscheinlich speziell entwickelte Font-Editoren benutzen. Die Art von Schrift, die Sie am häufigsten mittels Resource Workshop erstellen werden, sind Bilderschriften: kleine Bitmaps, die Sie zusammenfassen wollen.



In einigen Fällen sind Schrift-Ressourcen und Bitmap-Ressourcen nicht austauschbar. Wenn Sie ein Rastermuster erstellen, um eine Fläche auf dem Bildschirm auszufüllen, müssen Sie Bitmap-Ressourcen verwenden. Bei der Erstellung von Bitmap-Bildern, wie einer Bombe oder einem Stoppzeichen, für die Anzeige auf dem Bildschirm, können Sie aber sowohl Bitmap-, als auch Schrift-Ressourcen verwenden. Die Bombe am Rand wurde zum Beispiel als Schrift-Ressource erstellt.

Es gibt einige Gründe, Bilder als Teile einer Schrift-Ressource zu definieren:

- Es ist einfacher, Windows-Code zu schreiben, der eine Schrift in den Speicher lädt und ausgibt. Das Laden und Zeichnen des gleichen als Bitmap gespeicherten Bildes ist schwieriger.
- Eine Schrift kann bis zu 256 Bitmaps enthalten. Benutzt Ihr Programm häufig einen bestimmten Satz von Symbolen, so können Sie diese in einer Schrift-Ressource hinterlegen und Ihr Programm lädt die Ressource nur, wenn es den entsprechenden Satz benötigt.
- Eine Schrift kann eine Menge Bilder effizienter speichern, als dies in einer Menge einzelner Bitmap-Ressourcen möglich ist.

Eine Schrift-Ressource hat jedoch einen bestimmten Zusatzaufwand an Speicher für die Schriftbeschreibung, der bei jedem Laden der Schrift durch Windows anfällt. Bei der Erstellung eines einzelnen Bildes spricht das Kriterium der Speichereffizienz also für eine Bitmap-Ressource.

Erstellen einer neuen Schrift-Ressource

Sie können eine neue Schrift-Ressource einer Projektdatei zuordnen oder eine eigenständige .FNT-Datei erzeugen.

Zuordnen einer Schrift zu einer Projektdatei

Eine Schrift einer .RC (oder .DLG) -Datei zuordnen:

1. öffnen Sie ein neues oder bereits existierendes Projekt.
2. wählen Sie *Ressource/Neu*. Resource Workshop baut das entsprechende Dialogfenster auf.
3. wählen Sie *FONT* aus der Auswahlliste *Ressourcentyp*.
 Der Name der aktuellen Projektdatei erscheint im Feld unter *Ressource einfügen in*. Sie können in dieser Liste blättern, um eine andere Datei zu wählen (sofern gelistet) oder auf *OK* klicken, um die angezeigte Projektdatei zu bestätigen.
4. Resource Workshop stellt Sie vor die Wahl die Schrift als Quelltext oder Microsoft kompatible Binärdatei zu erstellen.
 Eingebettete und verknüpfte Ressourcen werden auf Seite p203 besprochen.

 Wählen Sie *Quelle*, um eine in die Projektdatei eingebettete Schrift zu erzeugen. Resource Workshop startet daraufhin den Bitmap-Editor. Wählen Sie *Binär*, so erstellen Sie eine eigenständige .FNT-Datei.

Erstellen einer eigenständigen Schriftdatei

Mit den beiden folgenden Alternativen können Sie eine eigenständige Schriftdatei anlegen:

- Wählen Sie *Binär* im Dialogfenster, das Sie nach dem Zielformat fragt.
- Wählen Sie *Datei/Neues Projekt* und wählen in dem dann angezeigten Dialogfenster *.FNT* als Dateityp.

Um eine eigenständige Schriftdatei mit einer Projektdatei zu verknüpfen, öffnen Sie die Projektdatei und wählen daraufhin *Datei/dem Projekt hinzufügen*.

Die Option binäres Format

Wählen Sie als Format Binär, so zeigt Resource Workshop das Dialogfenster »Neue Ressourcendatei«, so daß Sie Ihrer Schriftdatei einen Namen und das Projekt, in dem diese benötigt wird, angeben können. Ist der gewählte Dateityp .FNT, so steht im Feld *Dateiname* *.FNT.

1. Geben Sie einen Namen für Ihre .FNT-Datei im Feld *Dateiname* ein. Vergewissern Sie sich, daß der Name des gewünschten Projektes im Auswahlfeld *Ein Verweis ...* erscheint. Um das Verzeichnis, in dem die Datei abgelegt wird zu ändern, setzen Sie diesen durch Doppelklicken

in der Auswahlliste *Verzeichnisse*, was die Anzeige des angewählten Pfades veranlaßt.

2. Haben Sie Ihre Auswahl beendet, so klicken Sie *OK*. Resource-Workshop zeigt ein Dialogfenster, in dem Sie gefragt werden ob die Datei, da sie noch nicht existiert, neu angelegt werden soll. Klicken Sie auf *Yes*.
3. Beim Verlassen des Dialogfensters *Neue Ressourcendatei*, startet Resource-Workshop den Bitmap-Editor.

Sie können auch einen Dateinamen aus der Dateien-Auswahl wählen. In diesem Fall veranlassen Sie das Überschreiben der Datei, bei entsprechender Antwort auf die Nachfrage von Resource Workshop.

Die Schrift Projektdatei

Wählen Sie *Datei/Neues Projekt* und markieren daraufhin .FNT, so legen Sie automatisch eine eigenständige Schriftdatei an. Der Bitmap-Editor wird nach *OK* sofort gestartet.

Bearbeiten einer Schrift-Ressource

Beim Öffnen einer neuen oder bereits erstellten Schrift im Bitmap-Editor werden die Buchstaben oder Bilder der Schrift am rechten Fensterrand angezeigt.

Der Bitmap-Editor lädt nach dem Start sofort das erste Bild der Schrift. Ein anderes Bild laden Sie, indem Sie es im rechten Bereich anklicken. Mit den Werkzeugen nehmen Sie die gewünschten Änderungen vor.

Definition und Hinzufügen von Zeichen zu einer Schrift

Nach der Neuanlage einer Schrift enthält diese nur ein Bild von 8x8 Bildpunkten. Normalerweise möchte man jedoch mehr als ein Zeichen in den Zeichensatz seiner Schrift aufnehmen. Möglicherweise wollen Sie auch eine andere Größe für Ihre Schriftzeichen festlegen. Zur Erledigung dieser Aufgaben benutzen Sie den Hauptmenüpunkt *Schrift/Schriftgröße*.

Festlegen der Schriftgröße

Mit folgenden Schritten legen Sie die Schriftgröße fest:

1. Wählen Sie zur Anzeige des Dialogfensters *Informationen über Schriftgröße* den Menüpunkt *Schrift/Schriftgröße*.
2. Legen Sie die Optionen nach Ihren Erfordernissen fest. Die Bilder einer Schrift-Ressource können von variabler oder fester Breite sein.

Auf den nächsten Seiten finden Sie weitere Informationen zu Schriften variabler Breite.

- Schriften mit Zeichen variabler Breite werden auch Proportionalschriften genannt. Der Buchstabe "m" ist in einer solchen Schrift beispielsweise wesentlich breiter als ein "i". Die meisten Bücher - so auch dieses - werden in Proportionalschrift gedruckt.

- Schriften, in denen alle Zeichen dieselbe Breite haben, nennt man auch »monospaced«. Schreibmaschinen benutzen meist Schriften fester Zeichenbreite.

Zeichen variabler Breite nehmen meist weniger Platz ein und sehen besser aus als Zeichen fester Breite. Vergleichen Sie in den beiden folgenden Zeilen besonders die Darstellung um den Buchstaben "i".

Minimum (Schrift variabler Breite)

`Minimum (Schrift fester Breite)`

Tabelle 22.1: Optionen zur Größe

Option	Beschreibung
Breite	Sollen alle Zeichen die gleiche Breite haben, geben Sie die Breite in Pixeln an. Soll die Breite variieren, geben Sie hier 0 (Null) ein und bestimmen eine maximale Breite.
Höhe	Geben Sie die Höhe der Schriftbilder in Pixeln an.
Durchschnittl. Breite	Resource Workshop berechnet eine durchschnittliche Breite für die Schriftbilder, wenn Sie für Breite 0 (variable Breite) angegeben war. Ansonsten stimmt durchschnittl. Breite mit Breite überein. Dieser Wert wird beim Öffnen des Dialogfensters berechnet, ändert sich also während der Veränderungen von Einstellungen nicht.
Max. Breite	Für Schriften variabler Breite geben Sie die maximale Breite in Pixeln an. Diese Option ist nur verfügbar, wenn Sie bei Breite 0 angegeben haben.
Zeichenbreiten anpassen	Schalten Sie diese Option ein, wenn sich die schon vorhandenen Bilder in der Größe ändern sollen, entsprechend den Werten, die Sie in diesem Dialogfenster für Höhe oder Breite eingeben.

Festlegen der Anzahl von Zeichen

Das Dialogfenster *Informationen über Schriftgröße* enthält mit *Erstes* und *Letztes* auch Felder um die Größe des Zeichensatzes einer Schrift festzulegen (siehe folgende Tabelle).

Haben Sie die Anzahl anfangs zu niedrig gewählt, so können Sie dies auch später noch in diesen Feldern korrigieren.

Zuordnen des Zeichensatzes

Wählen Sie einen Bereich dezimaler Codes. Jedem Zeichen Ihrer Schrift wird dann ein bestimmter Code innerhalb dieses Bereichs zugeordnet. Wollen Sie beispielsweise ein Schriftbild dem Zeichen "a" zuordnen, geben Sie den Dezimalcode 97 an. Das Bild selbst muß nicht das Zeichen "a" sein, außer Sie wünschen dies. In der Beispielschrift am Ende dieses Kapitels ist dem "a" eine Bombe zugeordnet. Das zweite Schriftbild wird dann dem Buchstaben "b" mit dem Dezimalcode 98 zugeordnet.

➡ Die zugeordneten ANSI-Werte sind innerhalb des Bereiches 0 bis 255 frei wählbar. Diese ANSI-Werte werden relevant, wenn Sie die Schrift laden und einzelne Zeichen darstellen wollen, da der Zugriff auf diese über den zugeordneten ANSI-Wert erfolgt, genau so als wollten Sie ASCII-Zeichen anzeigen.

Zur Zuordnung Ihres Zeichen- oder Bildersatzes zu Ihrer Schrift verwenden Sie die Optionen aus folgender Tabelle.

Tabelle 22.2: Optionen zum Zeichen

Option	Beschreibung
Erstes	Geben Sie den ANSI-Dezimalcode für das erste Bild Ihrer Schrift ein. Soll dieses beispielsweise dem "a" zugeordnet werden, geben Sie 97 ein.
Letztes	Geben Sie den ANSI-Dezimalcode für das letzte Bild Ihrer Schrift ein. Soll dieses beispielsweise dem "z" zugeordnet werden, geben Sie 122 ein.
Standard	Geben Sie einen ANSI-Dezimalcode für das Zeichen ein, das immer dann auszugeben ist, wenn versucht wird, ein Zeichen anzufordern, das in dieser Schrift nicht definiert ist. Der Standardwert muß innerhalb des durch Erstes und Letztes gesetzten Bereiches liegen. Haben Sie diese wie oben beschrieben festgelegt, so können Sie für Standard nicht den Wert 88 angeben.
Füller	Geben Sie den ANSI-Dezimalcode für das Zeichen ein, das intern dazu benutzt wird, Ausgabezeilen aufzufüllen. Der Wert von Füller muß ebenfalls im durch Erstes und Letztes gesetzten Bereich liegen.

Erstellen von Zeichen variabler Breite

Gehen Sie zur Erstellung von Bildern variabler Breite für Ihre Schrift folgendermaßen vor:

1. Wählen Sie *Schrift/Schriftgröße*.
2. Geben Sie im Feld *Breite* 0 (Null) ein.
3. Geben Sie im Feld *Max. Breite* diese Größe in Pixeln ein. Der eingetragene Wert gilt für alle Bilder der Schrift-Ressource

Setzen der Breite eines Zeichens oder Bildes

Nachdem Sie als Breite für Ihre Schrift 0 (Null) festgelegt haben, können Sie die Breite eines bestimmten Zeichens oder Bildes durch *Schrift/Zeichenbreite* bestimmen.

Haben Sie beispielsweise in *Schrift/Schriftgröße* für ihre Schrift aus Bildern folgende Größen eingestellt:

Breite	0
Höhe	32
Max. Breite	32

Unter Berücksichtigung dieser Größen, haben alle Bilder eine Höhe von 32 Pixeln und jedes Bild eine maximale Breite von ebenfalls 32 Pixeln. Wenn Sie ein bestimmtes Bild bearbeiten, können Sie mit *Schrift/Zeichenbreite* die Breite dieses Bildes im Dialogfenster *Zeichenbreite* definieren.

Benutzen Sie *Zeichenbreite anpassen*, um das Bild an die neue Größe anzupassen.

Im Feld *Breite* geben Sie einen Wert kleiner oder gleich der maximalen Breite an. Zusätzlich können Sie *Zeichenbreite anpassen* wählen, um das entsprechende Bild abhängig von der neuen Breite zu vergrößern oder zu verkleinern.

Definieren von Header-Informationen für eine Schrift

Jede Schrift-Ressource enthält einen Header mit den allgemeinen Informationen der Schrift, wie dem Namen und den Copyright-Informationen. Wenn Sie beispielsweise eine Schrift definieren, die aus alphanumerischen Zeichen besteht, definiert der Header Schrifttyp und Größe für alle Zeichen dieser Schrift.

Zur Definition des Headers zu einer Schrift führen Sie diese Schritte aus:

1. Laden Sie die Schrift in den Bitmap-Editor, indem Sie den Schriftnamen im Projektfenster zweimal anklicken oder *Ressource Neu* wählen, um eine neu Schrift-Ressource zu erstellen.
2. Wählen Sie im Bitmap-Editor *Schrift/Header*. Das Dialogfenster *Header Informationen der Schrift* erscheint.
3. Definieren Sie den Header unter Verwendung der Möglichkeiten folgender Tabelle.

Tabelle 22.3: Schrift-Header-Optionen

Option	Beschreibung
Schrift	Geben Sie den Namen Ihrer Schrift ein.
Gerät	Legen Sie einen Gerätenamen fest, der die Verwendbarkeit der Schrift nur auf einem bestimmten Gerät anzeigt.
Copyright	Geben Sie die Copyright-Informationen der Schrift ein.
Schriftversion	Version 2.00 wird immer unterstützt. Die Benutzung von Version 3.00 ist möglich, wenn eine Windows 3.x Anwendung im Protected Mode (Standard- oder 386er-Modus) auf einem 80386 (oder höher) Prozessor auf die Schrift zugreift.
Attribute	
- Kursiv	Die Schrift enthält kursive Zeichen.
- Unterstrichen	Die Schrift enthält unterstrichene Zeichen.
- Durchgestrichen	Die Schrift enthält durchgestrichene Zeichen.
- Proportional	Die Schrift hat variable Zeichenbreite.
- Strichstärke	Die Schrift wird normal (400) oder fett (700) gedruckt.
- Familie	Beschreibt die Schriftfamilie. Die gültigen Werte lauten: 0 Nicht berücksichtigen 3 Modern 1 Roman 4 Script 2 Swiss 5 Decorative
- Zeichensatz	Definiert den Zeichensatz. Der Wert kann zwischen 0 und 255 liegen, aber nur 0, 2 und 255 haben vordefinierte Bedeutungen: 0 ANSI, der standardmäßige Windows-Zeichensatz.

	2 Symbol, für mathematische und wissenschaftliche Formeln 255 OEM, ein maschinenspezifischer Zeichensatz
Größen	
- Horizontale Auflösung	Die horizontale Auflösung von Pixeln pro logischem Zoll auf Ihrem Bildschirm.
- Vertikale Auflösung	Die vertikale Auflösung von Pixeln pro logischem Zoll auf Ihrem Bildschirm.
- Punkte	Die Zeichengröße. Ein Punkt entspricht $^{1}/_{72}$ eines Zolls. Ein Zeichen wird von der Oberlänge bis zu der Unterlänge gemessen. Der hier eingegebene Wert sollte keinen Platz für diakritische Zeichen vorsehen.
- Interner Durchschuß	Der für diakritische Zeichen reservierte Platz in Pixeln.
- Externer Durchschuß	Der zusätzliche Platz zwischen den Zeilen in Pixeln.
- Oberlänge	Die Höhe der Zeichen über der Grundlinie in Pixeln.

Ändern von Größe und Attributen

Beim Bearbeiten einer Schrifft-Ressource können Sie mit drei Befehlen des Menüs *Schrift* Änderungen an den Schriftbildern vornehmen.

- *Header* definiert die Informationen, für Ihre Schrift-Ressource, wie Schriftversion, Schriftnamen, Copyright-Information usw. Weitere Informationen im vorhergehenden Abschnitt.
- *Schriftgröße* definiert den Zeichensatz dieser Schrift, Breite und Höhe aller Zeichen.
- *Zeichenbreite* gibt die Breite für ein bestimmtes Bild in einer Schrift-Ressource. Dieser Befehl ist ausschließlich für Schriften variabler Breite ("Breite" ist 0 unter *Schrift/Schriftgröße*) verfügbar.

K A P I T E L 23

Erzeugen von benutzerdefinierten Ressourcen

Zusätzlich zu den in den vorigen Kapiteln beschriebenen Ressourcen-Typen können Sie eigene Ressourcen definieren. Wenn Sie einen neuen Ressourcen-Typ erzeugt haben, können Sie eine beliebige Anzahl *benutzerdefinierter Ressourcen* dieses Typs in Ihrem Projekt verwenden.

Es mag angebracht sein, eigene Ressourcen-Typen zu definieren, um Daten zu erfassen, die nicht zu einem der Standard-Ressourcen-Typen passen. Wenn Sie zum Beispiel eine Zeichenketten-Ressource erzeugen wollen, die mehr als 255 Zeichen lang ist (in STRINGTABLE erlaubtes Maximum), können Sie einen eigenen Ressourcen-Typ definieren und damit solche längeren Zeichenketten speichern.

Mehr Informationen über Metadateien finden Sie in Charles Petzolds »Programming Windows«.

Sie können auch Metadateien in Ihrem Projekt als benutzerdefinierte Ressourcen verwenden. Eine Metadatei ist eine Art Bitmap (im Quellformat ist sie eine Ansammlung von GDI-(Graphics Device Interface)Aufrufen. Die Metadatei kann nicht nur einfacher skaliert werden und ist unabhängiger von den verwendeten Geräten als die Standard-Bitmap-Ressource, sondern benötigt meist auch weniger Speicherplatz als eine Bitmap-Ressource.

Wenn Sie eine neue Ressource definieren, können die Daten entweder als Teil der Ressourcendefinition in einer Projektdatei oder in einer gesonderten Datei gespeichert werden. Wie bei jeder Ressource, wird Resource Workshop die Daten compilieren und an die ausführbare Datei binden, so daß die Daten beim Programmstart für Ihre Anwendung verfügbar sind.

➥ Sie können auch den Ressourcen-Typ RCDATA verwenden, um Daten an Ihre Anwendung anzufügen.

Die Arbeit mit benutzerdefinierten Ressourcen erfolgt in fünf Schritten:

1. Erzeugen eines benutzerdefinierten Ressourcen-Typs.
2. Hinzufügen der benutzerdefinierten Ressource zu Ihrem Projekt.
3. Bearbeiten (d.h. Editieren) der benutzerdefinierten Ressource.
4. Testen der benutzerdefinierten Ressource.
5. Speichern der benutzerdefinierten Ressource.

Erzeugen eines Ressourcen-Typs

Bevor Sie eine benutzerdefinierte Ressource in Ihrem Projekt verwenden können, müssen Sie zuvor einen Typ für sie erzeugt haben. Folgendes muß dazu getan werden:

Wie Sie ein Projekt öffnen, steht in Kapitel 14.

1. Öffnen Sie ein Projekt.
2. Wählen Sie *Ressource/Neu*.
3. Im Dialogfenster *Neue Ressource* klicken Sie die Taste *Neuer Typ* an und das Dialogfenster *Neuer Ressourcen-Typ* erscheint.
4. Geben Sie im Dialogfenster *Neuer Ressourcen-Typ* den Namen des zu erzeugenden Ressourcen-Typs an. Wenn Sie z.B. eine Ressource definieren wollen, um einen überlangen Textblock aufzunehmen, können Sie den Ressourcen-Typ TEXT nennen. Drücken Sie die [*Eingabetaste*].

Ein numerischer Wert für die ID muß größer als 255 sein, da Windows die Werte 1 bis 255 für Standard-Ressourcen verwendet

5. Wenn die Anfrage »*Neuen Bezeichner erzeugen?*« erscheint, klicken Sie auf *Ja* und geben einen Wert für den Bezeichner in das Wertefeld des Dialogfensters *Neuer Bezeichner* ein. Dieser Wert ist die ID, mit dem Windows und Ihr Programm diesen Bezeichnertyp eindeutig erkennen.

Wenn Sie erst einen neuen Ressourcen-Typ definiert haben, wird dieser Ressourcen-Typ mit all den anderen Standard-Ressourcen-Typen (ACCELERATOR, BITMAP, DIALOG usw.) im Dialogfenster *Neue Ressource* aufgelistet, sobald Sie eine neue Ressource erzeugen wollen.

Hinzufügen benutzerdefinierter Ressourcen

Nachdem Sie einen Ressourcen-Typ erzeugt haben, können Sie eine Ressource dieses Typs Ihrem Projekt wie folgt zufügen:

Wie Sie ein Projekt öffnen, steht in Kapitel 14.

1. Öffnen Sie ein Projekt.
2. Wählen Sie *Ressource/Neu*. Resource Workshop zeigt das Dialogfenster *Neue Ressource*.

3. Wählen Sie im *Neue Ressource*-Dialogfenster Ihren benutzerdefinierten Ressourcen-Typ aus. Unter dem Punkt »*Ressource einfügen in*« wählen Sie die Projektdatei (dies wird wahrscheinlich die aktuelle Datei sein), in der Sie die Ressource speichern wollen.
4. Wenn Sie ins Dialogfenster *Neue Ressource* klicken, ruft Resource Workshop den Texteditor mit einer Definitionsvorlage für Ihre benutzerdefinierte Ressource auf. Wenn Sie z.B. eine benutzerdefinierte Ressource namens TEXT hinzufügen, sehen Sie im Texteditor den folgenden Quelltext:

```
TEXT_1 TEXT
{

}
```

Der nächste Abschnitt beschreibt, wie Sie eine benutzerdefinierte Ressource bearbeiten.

Bearbeiten von benutzerdefinierten Ressourcen

Wie Sie ein Projekt öffnen, steht in Kapitel 14.

Um eine benutzerdefinierte Ressource zu bearbeiten, muß ein Projekt geöffnet sein. Sie können entweder eine Ressource erzeugen (siehe den vorigen Abschnitt) oder Sie können eine bereits vorhandene im Projektfenster öffnen.

Der Name der Ressource ist im Projektfenster unter dem benutzerdefinierten Ressourcen-Typ aufgelistet, genauso wie jede beliebige Ressource eines vordefinierten Typs aufgelistet ist. Wenn Sie zum Beispiel einen Ressourcen-Typ TEXT und eine Ressource TEXT erzeugen, sehen Sie den Ressourceneintrag TEXT im Projektfenster.

Um den Editor aufzurufen,

- Doppelklicken Sie auf den Namen der Ressource, die Sie bearbeiten wollen oder
- Markieren Sie den Ressourcennamen und wählen Sie dann *Ressource/Bearbeiten* oder *Ressource/Bearbeiten als Text*. (Beides startet den Editor.)

Wenn Sie die Ressource in den Texteditor geladen haben, können Sie weitere Daten hinzufügen oder die vorhandenen Daten bearbeiten. Folgendes wird z.B. angezeigt, wenn Sie Ihrem Projekt eine neue benutzerdefinierte Ressource des Typs RESTYPE hinzufügen:

```
RESTYPE_1 RESTYPE
BEGIN
 [Definition der Daten]
END
```

Die erste Zeile enthält den Ressourcennamen und den Typ. Resource Workshop erzeugt einen Standardnamen für eine neue Ressource, indem es an den Ressourcen-Typnamen einen Unterstrich und eine Nummer angehängt. Die erste Ressource des Typs RESTYPE zum Beispiel, die Sie Ihrem Projekt zufügen, wird RESTYPE_1 genannt, die zweite heißt RESTYPE_2, und so weiter.

Um Daten Ihrer Ressource hinzuzufügen, wählen Sie eine der folgenden Vorgehensweisen

- Benutzen Sie den Texteditor, um Daten in die geschweiften Klammern einzugeben.
- Speichern Sie die Daten in einer gesonderten Datei und fügen Sie den Dateinamen am Ende der ersten Zeile des Ressourcenskripts wie folgt ein:

  ```
  RESTYPE_1 RESTYPE MYRES.FIL
  ```

 Sie müssen auch die Zeilen mit den BEGIN- und END-Anweisungen löschen.
- Benutzen Sie die Anweisung *Dem Projekt hinzufügen*, die im nächsten Abschnitt beschrieben wird.

Wenn Sie Änderungen im Ressourcenskript gemacht haben, müssen Sie die Ressource erneut compilieren lassen, um die Änderungen zu speichern. Wenn Sie den Vorgang ohne erneutes Compilieren abbrechen, sind alle Änderungen verloren.

Hinzufügen von Ressourcendaten in die Projektdatei

Wie im letzen Abschnitt dargestellt, können Sie Ihrer Ressource Daten hinzufügen, indem Sie die Daten in einer gesonderten Datei speichern. Der Nachteil dieses Vorgehens ist, daß die Daten verloren sind, falls die Datei beschädigt wird. Eine andere Möglichkeit besteht darin, die externen Daten in das Projektdateiskript einzufügen.

Haben Sie zum Beispiel einen benutzerdefinierten Ressourcen-Typ namens TEXT und eine Ressourcendatei mit Namen MY_RES.TXT, können Sie diese Daten in das Projekt wie folgt einfügen:

1. Wählen Sie *Datei/Dem Projekt hinzufügen*. Resource Workshop zeigt das Dialogfenster *Datei einem Projekt hinzufügen*.
2. Markieren Sie »*Benutzerdefinierte Ressourcendaten*« im Listenfeld Dateityp.
3. Ändern Sie das Dateinamen Eingabefeld in `*.TXT`
4. Wechseln Sie das Verzeichnis (falls notwendig), bis MY_RES.TXT in der Dateiliste auftaucht. Doppelklicken Sie auf MY_RES.TXT.

Sie können auch den vollständigen Pfad und Dateinamen im Eingabefeld Dateinamen angeben

5. Resource Workshop zeigt das Dialogfenster *Benutzerdefinierter Ressourcen-Typ*. Doppelklicken Sie auf den Eintrag für den Ressourcen-Typ TEXT. Eine neue TEXT-Ressource erscheint im Projektfenster.

Wenn Sie die neue Ressource markieren und *Ressource/Bearbeiten* (oder *Bearbeiten als Text* - das Ergebnis ist dasselbe) auswählen, werden Sie sehen, daß die Ressourcendaten in eine hexadezimale Darstellung umgewandelt worden sind. Das ist der Grund, warum Sie die externe Datendatei aufbewahren sollten für den Fall, daß Sie das Ressourcenskript später noch einmal bearbeiten wollen.

Daten ins Ressourcenskript eingeben

Im Texteditor verwenden Sie [*Tab*], [*Entf*], [*Pos1*], [*Ende*], [*Nach oben*], [*Nach unten*] und [*Rück*] wie gewohnt. Verwenden Sie den Editor jedoch nicht, um den Text zu formatieren, da Resource Workshop den Textaufbau beim Compilieren oder Decompilieren der Ressource wahrscheinlich ohnehin verändert.

Einige Richtlinien, wie Daten zwischen den Parametern BEGIN und END eingegeben werden sollten:

- Die Daten können numerische Werte und Zeichenketten in beliebiger Kombination umfassen.
 - Sie können Zahlen in hexadezimaler, oktaler oder dezimaler Schreibweise angeben.
 - Benutzen Sie entweder 0x (eine Null gefolgt vom Buchstaben x) oder $ (das Dollarzeichen) als erste Zeichen bei hexadezimaler Schreibweise. Bei dieser Schreibweise werden nur 16-Bit-Werte unterstützt. Wenn Sie eine ungerade Anzahl hexadezimaler Werte verwenden wollen, benutzen Sie den Datentyp *hexstring* (im nächsten Unterpunkt beschrieben)
- Verwenden Sie 0o (eine Null gefolgt vom Buchstaben o) oder einfach 0 (Null) als führende Zeichen bei oktaler Schreibweise.
 Sie können hexadezimale Werte auch darstellen, indem Sie eine *hexstring*-Zeichenkette verwenden. Diese ist eine Folge von hexadezimalen Werten, die durch einfache Anführungsstriche begrenzt wird. Der Compiler überliest alle Leerzeichen, die Sie einfügen, um den hexadezimalen Code besser lesbar zu gestalten.
 Zum Beispiel können Sie die ASCII-Werte der Buchstaben im Wort *string* als `'73 74 72 69 6E 67'` und die Hexadezimalzahl 06E07A als `'06E07A'` schreiben.
- Wenn Sie Textzeichenketten in Ihren Ressourcen verwenden, begrenzen Sie die Zeichenkette mit gewöhnlichen Anführungsstrichen, wie "*string*".

Zeichenketten werden nicht automatisch mit einer 0 abgeschlossen. Um eine Zeichenkette mit einem Nullzeichen zu beenden, geben Sie \0 (Backslash 0) am Ende der Zeichenkette ein.

Ihr Ressourcenscript mit eingefügten Daten könnte z.B. wie folgt aussehen:

```
RESTYPE_1 RESTYPE
BEGIN
  "This is a string."
  0xFFAA 0o7076 01077
  '54 68 69 73 20 69 73 0D 0A 73 6F 6D 65 20 73 61'
  '6D 70 6C 65 0D 64 61 74 61 2E'
END
```

Verwaltung von Daten aus gesonderten Dateien

Wenn die Daten in einer gesonderten Datei gespeichert sind, müssen Sie einen externen Editor benutzen, um die Datei zu bearbeiten. Um auf die Datendatei im Ressourcenskript zu verweisen,

- Geben Sie den vollständigen Pfadnamen in der ersten Zeile des Skripts direkt nach dem Namen des Ressourcen-Typs an.
- Löschen Sie die BEGIN-und END-Anweisungen.

Das folgende Ressourcenskript für die Ressource RESTYPE_1 zeigt, daß die Daten in der Datei C:\RW\MYDATA.TXT gespeichert sind:

```
RESTYPE_1 RESTYPE c:\rw\mydata.txt
```

Wie Sie den Ressourcen-Typ RCDATA verwenden

Der Typ RCDATA wurde in Resource Workshop aus Kompatibiliätsgründen eingefügt.

Sie können den vordefinierten Ressourcen-Typ RCDATA verwenden, um Ihrer Anwendung eine Datenressource hinzuzufügen. Dies funktioniert genauso wie bei einem benutzerdefinierten Ressourcen-Typ. Der hauptsächliche Unterschied zwischen den beiden Möglichkeiten liegt in der differenzierteren Ansprechmöglichkeit bei benutzerdefinierten Typen. Es kann für Sie vorteilhafter sein, verschiedene Typen benutzerdefinierter Ressourcen zu haben als nur einen einzigen Typ, RCDATA.

Wenn Sie eine RCDATA-Ressource verwenden wollen, nehmen Sie sie in das Projekt auf, indem Sie *Ressource/Neu* wählen und eine neue Ressource des Typs RCDDATA einfügen. Sie sehen eine unvollständige RCDATA-Definition im Texteditor. Nun können Sie Daten zwischen die BEGIN- und END-Anweisungen einfügen. Für die Eingabe der Daten gelten dieselben Regeln wie bei benutzerdefinierten Ressourcen.

A N H A N G **A**

Fehlermeldungen

Dieser Anhang befaßt sich mit den Fehlermeldungen, die von Borland C++ erzeugt werden können. Dazu gehören die Fehlermeldungen des Compilers, des Dienstprogrammes MAKE, der Bibliothek (TLIB), des Linkers (TLINK) sowie des Hilfe-Compilers. Zusätzlich enthält dieser Anhang die Fehlermeldungen, die zur Laufzeit des Programms entstehen können, sowie die Laufzeitfehler.

Die Meldungen sind in ASCII-Reihenfolge angeordnet. Die Meldungen können mit Symbolen, Zahlen oder Buchstaben beginnen. Die Meldungen mit Symbolen werden als erste angezeigt; dann folgen die Meldungen mit Zahlen und zuletzt die Meldungen mit Buchstaben. Meldungen, die mit einem Symbol beginnen, sind nach dem ersten dem Symbol folgenden Wort geordnet. Beispielsweise können Sie die folgende Fehlermeldung erhalten, wenn Sie eine Funktion *my_func* falsch deklariert haben.

```
my_func must be declared with no parameters
```

Diese Fehlermeldung finden Sie, wenn Sie im alphabetischen Verzeichnis unter »must« nachschlagen.

Fehlerklassen

Die Fehlerklassen werden in drei Kategorien unterteilt: schwerwiegende Fehler, Fehler und Warnungen

Schwerwiegende Fehler

Schwerwiegende Fehler können durch den Compiler, den Linker oder durch das Hilfsprogramm MAKE hervorgerufen werden. Sie führen zu einem sofortigen Abbruch der Compilierung. Sie müssen dann den Fehler korrigieren und die Compilierung erneut starten. Wenn der Compiler oder das Hilfsprogramm MAKE einen schwerwiegenden Fehler ausgibt,

wird keine EXE-Datei erzeugt. Gibt der Linker einen schwerwiegenden Fehler aus, so wird jede EXE-Datei die bereits erzeugt wurde, gelöscht.

Fehler

Fehler können vom Compiler, vom Linker, vom Hilfsprogramm MAKE oder von der Bibliothek erzeugt werden. Außerdem können Fehler auch zur Laufzeit von Ihren eigenen Programmen erzeugt werden.

Fehler, die vom Compiler erzeugt werden, deuten auf syntaktisch inkorrekte Befehle, auf Fehler in Kommandozeileneingaben sowie auf fehlerhafte Platten- oder Speicherzugriffe hin. Compiler-Fehler beenden das Compilieren nicht. Der Compiler beendet nur die momentane Compilationsphase; erst danach hält er an, um die aufgetretenen Fehler zu melden. Der Compiler versucht, möglichst alle wirklichen Fehler während einer Compilierungsphase (Präprozessorphase, Parsing, Optimierung und Code-Erzeugung) im Quelltext zu finden.

Fehler, die durch den Linker generiert werden, führen nicht zur Löschung der EXE- und MAP-Dateien. Jedoch sollten Sie keine EXE-Dateien ausführen, die mit Fehlermeldungen gelinkt wurden. Linkerfehler werden wie schwerwiegende Fehler behandelt, wenn Sie aus der Entwicklungsumgebung compiliert haben.

Das Hilfsprogramm MAKE generiert Fehlermeldungen, wenn in der Quell-MAKE-Datei syntaktische oder semantische Fehler aufgetreten sind. Sie müssen die Makedatei überprüfen, um diese Fehler zu korrigieren.

Laufzeitfehler werden gewöhnlich durch logische Fehler im Programmcode hervorgerufen. Wenn Sie einen Laufzeitfehler erhalten, müssen Sie den Fehler im Quelltext beheben. Danach muß das Programm neu compiliert werden, damit die Korrektur berücksichtigt wird.

Warnungen

Warnungen können durch den Compiler, den Linker oder die Bibliothek verursacht werden. Sie führen nicht zum Abbruch der Compilierung, sondern geben Hinweise auf Befehlsfolgen, die dem Compiler »verdächtig« vorkommen, aber nicht gegen die Sprachregeln verstoßen. Der Compiler gibt Warnungen auch dann aus, wenn Sie maschinenabhängige Konstrukte in Ihren Quelltextdateien verwenden.

Meldungen des Hilfe-Compilers

Der Hilfe-Compiler zeigt Fehler oder Warnungen an, wenn er bei der Erstellung der Hilfe-Ressourcen-Datei auf Fehler stößt. Meldungen, die bei der Abarbeitung der Projektdatei generiert werden, bestehen aus dem Buchstaben **P** und der Fehlernummer. Meldungen, die bei der Abarbei-

tung der RTF-Themendatei auftreten, bestehen aus dem Buchstaben **R** und der Fehlernummer.

Wenn möglich, wird der Hilfe-Compiler die Themennummer und/oder den Dateinamen ausgeben, wo der Fehler aufgetreten ist. Wenn Sie Ihre Themen numerieren, wird die Themennummer mit einer Meldung ausgegeben, die sich auf die Reihenfolge innerhalb der RTF-Datei bezieht (erstes Thema, zweites Thema usw.). Diese Nummern können mit der Seitennummer identisch sein, die von Ihrer Textverarbeitung angezeigt wird. In Ihrer Hilfe-Quelldatei sind die Themen durch harte Seitenumbrüche getrennt, obwohl es in dem Hilfe-System keine »Seiten« gibt.

Meldungen, die mit dem Wort »Error« beginnen, sind schwerwiegende Fehler. Sie werden immer ausgegeben. Es wird dabei jedoch keine korrekte Ressourcen-Datei erzeugt. Meldungen, die mit dem Wort »Warning« beginnen, sind in der Regel weniger schwerwiegend. Eine Hilfe-Ressourcen-Datei, die Warnungen enthält, ist gültig, kann jedoch organisatorische Fehler enthalten.

Meldungslisten

Bei Meldungen wird die Fehlerklasse zuerst ausgegeben. Danach folgt der Quelldateiname, die Nummer der Zeile, in der der Fehler auftrat, und zum Schluß der eigentliche Text der Meldung.

Einige Meldungen enthalten ein Symbol (z.B. eine Variable, einen Dateinamen oder den Namen eines Programmmoduls) aus Ihrem Programm. Die Symbole in den Erläuterungen zu den Meldungen sind *kursiv* gesetzt, damit sie als Variablen erkannt werden können.

Bitte beachten Sie, daß der Compiler Meldungen generiert, sobald er sie entdeckt. Da sich in C und in C++ eine Anweisung über mehrere Zeilen erstrecken kann, kann die vom Compiler gemeldete Zeilennummer von der Nummer der Zeile abweichen, in der der Fehler tatsächlich aufgetreten ist.

Erläuterung der Meldungen

) missing in macro invocation *MAKE-Fehler*

) fehlt in dem Makroaufruf

Die rechte Klammer fehlt im Makroaufruf.

(expected *Compiler-Fehler*

(erwartet

Vor einer Parameterliste wird eine öffnende, runde Klammer erwartet.

) expected *Compiler-Fehler*

) erwartet
Am Ende einer Parameterliste wird eine abschließende rechte Klammer erwartet.

, expected *Compiler-Fehler*

, erwartet
Komma fehlt

: expected after private/protected/public *Compiler-Fehler*

: nach **private protected public** erwartet
Am Beginn eines entsprechenden Abschnitts einer C++ Klasse muß auf die reservierten Begriffe **private**, **protected** und **public** ein Doppelpunkt folgen.

< expected *Compiler-Fehler*

< erwartet
Dem Schlüsselwort **template** folgt kein <.
In jeder Template-Deklaration muß nach dem Schlüsselwort template die Liste der formalen Parameter, eingeschlossen in < >, folgen.

> expected *Compiler-Fehler*

> erwartet
Bei einer **new**-Typ-Umwandlung (z.B.: dynamic_cast) fehlt das abschließende **>** Zeichen.

@ seen, expected a response-files name *Bibliotheken-Fehler*

@ gefunden, Response-Dateiname erwartet
Der Name der Response-Datei wurde nicht unmittelbar nach dem »Klammeraffen« @ angegeben.

{ expected *Compiler-Fehler*

{ erwartet
Beim Beginn eines Blocks oder einer Initialisierung wird **{** erwartet.

} expected *Compiler-Fehler*

} erwartet
Am Ende eines Blocks oder einer Initialisierung wird eine abschließende rechte Klammer erwartet.

16-bit segments not supported in module *Modulname* *Linker-Fehler*

16-Bit-Segment in Modul *Modulname* wird nicht unterstützt
16-Bit-Segmente werden für Win32-Anwendungen nicht unterstützt. Überprüfen Sie, ob Sie die Anwendung nicht mit dem 16-Bit-Compiler übersetzt haben.

286/287 instructions not enabled *Compiler-Fehler*

286/287-Instruktionen nicht aktiviert
Verwenden Sie die Kommandozeilenoption **-2** oder die 80286-Option im Dialogfenster *Optionen/Projekt/Compiler*, um die Generierung von 286-/287-Opcode zu aktivieren. Dabei müssen Sie bedenken, daß der erzeugte Programmcode nicht auf Rechnern mit 8086- bzw. 8088-Prozessoren ausgeführt werden kann.

32-bit record encountered *Linker-Fehler*

32-Bit-Record vorhanden
Eine Objektdatei mit 80386 32-Bit-Sätzen wurde entdeckt, obwohl die Option **/3** nicht benutzt wurde.

Abnormal program termination *Laufzeit-Fehler*

Außergewöhnliches Programmende
Da dem Programm nicht genügend Speicher zur Ausführung zur Verfügung stand, wurde *abort* aufgerufen. Diese Meldung kann durch Überschreiben des Speichers verursacht werden.

Access can only be changed to public or protected *Compiler-Fehler*

Zugriff nur in **public** oder **protected** veränderbar

Eine abgeleitete Klasse in C++ kann die Zugriffsrechte eines Basiselements der Basisklasse nur in **public** oder **protected** verändern. Ein Element einer Basisklasse kann nicht in **private** verändert werden.

Added file *Dateiname* **does not begin correctly, ignored** *Bibliotheken-Warnung*

Hinzugefügte Datei *Dateiname* beginnt falsch und wurde ignoriert

TLIB hat festgestellt, daß es sich bei der hinzuzufügenden Datei nicht um ein Objektmodul handelt. Es wird nicht versucht, die Datei der Bibliothek hinzuzufügen. Die Bibliothek wird dennoch erzeugt.

Address of overloaded function *Funktion* **doesn't match** *Typ* *Compiler-Fehler*

Die Adresse der überladenen Funktion *Funktion* paßt nicht zum Typ *Typ*

Einer Variable oder einem Parameter wurde (evtl. bei der Initialisierung) die Adresse einer überladenen Funktion zugewiesen. Der Typ der Variable oder des Parameters paßt jedoch zu keiner der überladenen Funktionen mit dem angegebenen Namen.

Modulname **already in LIB, not changed!** *Bibliotheken-Warnung*

Modulname schon in LIB, keine Änderung erfolgt

Es wurde der Versuch unternommen, die **+** Aktion auf der Bibliothek auszuführen, aber ein Objekt mit gleichem Namen existiert bereits in der Bibliothek. Wird eine Änderung dieses Objektes gewünscht, so sollte die Aktion **+-** verwendet werden. Die Bibliothek wurde nicht verändert.

Ambiguity between *Funktion1* **and** *Funktion2* *Compiler-Fehler*

Funktion1 und *Funktion2* mehrdeutig

Beide überladenen Funktionen können über die angegebenen Parameter angesprochen werden. Diese Mehrdeutigkeit ist nicht zulässig.

Ambiguous member name *Name* *Compiler-Fehler*

Zweideutiger Elementname *Name*

Wenn der Name eines Strukturelementes in einer inline-Assembleranweisung vorkommt, muß dieser Name eindeutig sein. Kommt der Name in mehreren Strukturen vor, so müssen alle Vorkommen sowohl im Datentyp als auch in der Position innerhalb der Struktur übereinstimmen. Hier wurde ein nicht eindeutiger Elementname verwendet. Benutzen Sie stattdessen:

```
(struct xxx).yyy
```

Ambiguous Override of Virtual Base Member *Funktion1*: *Funktion2* *Compiler-Fehler*

Mehrdeutiges Überschreiben von Elementfunktionen virtueller Basisklassen *Funktion1*:*Funktion2*

Eine virtuelle Funktion einer virtuellen Basisklasse wurde von zwei oder mehreren verschiedenen Funktionen entlang verschiedener Wege der Vererbungshierarchie überschrieben.

Ambiguous operators need parentheses *Compiler-Fehler*

Mehrdeutige Operatorkombinationen ohne Klammerung

Diese Warnung erhalten Sie bei der Kombination von relationalen, bitweisen oder Schiebe-Operatoren ohne Klammerung oder bei der Kombination eines Additions- oder Subtraktionsoperators mit einem Schiebeoperator ohne Klammerung. Programmierer verwechseln häufig die Rangfolge dieser Operatoren, da die ihnen zugeordnete Rangfolge der intuitiven Annahme widerspricht.

Ambiguous override of Virtual Base Member *Basis_Funktion1*:*Abgeleitete_Funktion2* *Compiler-Fehler*

Mehrdeutiges Überschreiben von Elementfunktionen virtueller Basisklassen *Basis_Funktion1*:*Abgeleitete_Funktion2*

Eine virtuelle Funktion einer virtuellen Basisklasse wurde von zwei oder mehreren verschiedenen Funktionen entlang verschiedener Wege der Vererbungshierarchie überschrieben. Zum Beispiel:

```
struct VB
{
   virtual f();
};

struct A:virtual VB
{
   virtual f();
};

struct B:virtual VB
{
   virtual f();
};

struct D:A,B
{
       // hier sind die Fehler
};
```

Dieser Text würde folgende Fehlermeldungen verursachen:

```
Error: Ambiguous override of virtual base member VB::f():A::f()
Error: Ambiguous override of virtual base member VB::f():B::f()
```

Application load & execute error 0001 *Compiler-Fehler*
Application load & execute error FFE0

Anwendungs Lade- und Ausführungsfehler FFE0 (bzw. 0001)
Es ist nicht genügend Erweiterungsspeicher vorhanden, um die Protected-Mode-Kommandozeile zu laden.

Array allocated using new may not have an initializer *Compiler-Fehler*

Das mit **new** reservierte Array darf keinen Initialisierer verwenden
Beim Initialisieren eines Vektors (Arrays) von Klassen muß ein Konstruktor ohne Argumente verwendet werden. Dieser Konstruktor heißt *Standard-Konstruktor*. Das bedeutet, daß bei der Initialisierung eines solchen Vektors keine Konstruktor-Argumente angegeben werden dürfen.

Array bounds missing] *Compiler-Fehler*

] bei Array-Deklaration oder Adressierung erwartet
In Ihrem Quelltext ist ein Array deklariert, dessen rechte, eckige Klammer fehlt.

Array must have at least one element *Compiler-Fehler*

Array muß mindestens ein Element enthalten
ANSI C und C++ schreiben vor, daß ein Array mindestens ein Element enthalten muß, da Objekte der Größe Null nicht erlaubt sind. Ein alter Programmiertrick sieht als Element einer Struktur ein Array der Größe Null vor und liefert später den nötigen Speicherplatz mit **malloc**. Um diesen Trick nach wie vor anzuwenden, müssen Sie mindestens ein Element für das Array vorsehen, wenn Sie im ANSI-Standard arbeiten. Im Gegensatz zu Definitionen sind weiterhin Deklarationen von Arrays unbekannter Größe möglich, wie z.B:

```
char ray[]; /* Definition unbekannter Größe – NICHT ERLAUBT */
char ray[0]; /* Definition der Größe Null – NICHT ERLAUBT*/
extern char ray[]; /* Deklaration unbekannter Größe – ERLAUBT */
```

Array of references is not allowed *Compiler-Fehler*

Array aus Referenzen nicht erlaubt
Ein Array aus Referenzen ist nicht zulässig, weil Zeiger auf Referenzen nicht erlaubt sind und Arraynamen in Zeiger umgewandelt werden.

Array size for delete ignored *Compiler-Warnung*

Array-Größe für **delete** wird ignoriert

In der neuen Version von C++ ist die Angabe der Array-Größe beim Löschen des Array nicht mehr nötig. Der Compiler ignoriert diese Angabe und gibt die vorliegende Warnung aus, wenn Sie ältere Quelltexte compilieren.

Array size too large *Compiler-Fehler*

Array zu groß

Das deklarierte Array ist größer als 64 KByte.

Array variable *Bezeichner* **is near** *Compiler-Warnung*

Array-Variable *Bezeichner* ist vom Typ **near**

Bei Verwendung der Kommandozeilenoption **-Ff** oder **-Fm** bzw. der Auswahl von *Optionen/Projekt/16-Bit Compiler/Speichermodell* definiert der Compiler globale Variablen, die größer als der Schwellenwert sind, automatisch als **far**. Ist die Variable ein initialisiertes Array von unbekannter Größe, dann ist dem Compiler die Gesamtgröße nicht bekannt, wenn er über die Verwendung von **near** oder **far** entscheiden muß; in diesem Fall wird **near** angenommen. Der Compiler gibt diese Warnung aus, wenn durch die Anzahl der für das Array gegebenen Initialisierer die Gesamtgröße der Variablen die Schwellengröße überschreitet. Erklären Sie die betreffende Variable explizit als **far**, falls durch die Festlegung der Variablen durch den Compiler als **near** Probleme auftreten. Dazu fügen Sie das Schlüsselwort **far** unmittelbar vor dem Variablennamen in der Definition ein.

Assembler statement too long *Compiler-Fehler*

Assembler-Anweisung zu lang

Inline-Assembler-Anweisungen dürfen nicht mehr als 480 Bytes umfassen.

Assigning *Typ* **to** *enumeration* *Compiler-Warnung*

Zuweisung von *Typ* an *enumeration*

Es ist ein Fehler in C++, einem **enum**-Typ einen Integerwert zuzuweisen. Hier wird aber nur eine Warnung ausgegeben, damit existierende Programme laufen können.

Assignment to this not allowed, use X::operator new instead *Compiler-Fehler*

Zuweisung an **this** nicht erlaubt, **::***operator new* verwenden

In früheren Versionen von C++ war die einzige Möglichkeit, die Speicherbelegung einer Klasse von Objekten zu überwachen, die Zuweisung an den Parameter **this** innerhalb eines Konstruktors. Dies ist nicht mehr erlaubt. Eine bessere, sicherere und allgemein anwendbare Möglichkeit besteht darin, eine Elementfunktion **operator new** zu definieren.

Attempt to export non-public symbol *Symbol* *Linker-Fehler*

Versuch ein als nicht **public** deklariertes Symbol *Symbol* zu exportieren

Dieser Fehler tritt gewöhnlich auf, wenn in einer DEF-Datei ein Export für ein nicht definiertes Symbol gefunden wurde. Vielleicht haben Sie das Symbol *Symbol* falsch geschrieben.

Attempt to grant or reduce access to *Bezeichner* *Compiler-Fehler*

Zugriffsrechte auf *Bezeichner* können nicht geändert werden

In C++ kann eine abgeleitete Klasse die Zugriffsrechte eines Basisklassenelementes abändern, aber nur, um sie nachher wieder herzustellen. Es können keine Rechte hinzugefügt oder weggenommen werden.

Attempting to return a reference to a local object *Compiler-Fehler*

Rückgabe einer Referenz auf ein lokales Objekt

Sie versuchen, eine Referenz auf ein temporäres Objekt in einer Funktion zurückzugeben, die einen Referenztyp liefert. Dies kann das Ergebnis eines Konstruktors oder eines Funktionsaufrufes sein. Bei Funktionsende verschwindet dieses Objekt, wodurch die Referenz unzulässig wird.

Attempting to return a reference to local variable *Bezeichner* *Compiler-Fehler*

Rückgabe einer Referenz auf die lokale Variable *Bezeichner* versucht

Diese C++ Funktion liefert eine Referenz, die sich auf eine lokale (auto) Variable beziehen soll. Dies ist nicht zulässig, weil die lokale Variable beim Verlassen der Funktion verschwindet. Eine Referenz darf sich nur auf eine statische oder globale Variable beziehen, oder die Funktion muß einen Wert anstatt einer Referenz liefern.

Bad call of intrinsic function *Compiler-Fehler*

Unzulässiger Aufruf einer **intrinsic**-Funktion

Sie haben eine **intrinsic**-Funktion verwendet, ohne einen Prototyp bereitzustellen, oder Sie haben einen Prototyp für eine **intrinsic**-Funktion angegeben, den der Compiler nicht erwartet hat.

Bad character in parameters *Linker-Fehler*

Unzulässiges Zeichen in der Parameterliste

Eines der folgenden Zeichen trat in der Kommandozeile oder in einer Antwort-Datei auf:

`" * < = > ? [ ]` (oder ein beliebiges Steuerzeichen außer Horizontal-Tabulator, Zeilenvorschub, Wagenrücklauf oder [*Strg*] [*Z*]).

Bad define directive syntax *Compiler-Fehler*

Ungültige **#define**-Direktive

Eine Makro-Definition beginnt oder endet mit einem ##-Operator, oder sie enthält einen #-Operator ohne darauffolgendes Makro-Argument.

Bad field list in debug information in module *Modulname* *Linker-Fehler*

Ungültige Feldliste in der Debug-Information in Modul *Modulname*

Dieser Fehler wird typischerweise durch ungültige Debug-Informationen in der OBJ-Datei verursacht. Sie sollten Borland verständigen.

Bad file name *Dateiname* *Linker-Fehler*

Ungültiger Dateiname *Dateiname*

Dem Linker wurde ein ungültiger Dateiname übergeben.

Bad file name format in include directive *Compiler-Fehler*

Ungültiger Dateiname in **#include**-Anweisung

Dateinamen müssen entweder mit Anführungszeichen ("filename.h") oder mit spitzen Klammern (<filename.h>) eingefaßt sein. Dem angegebenen Dateinamen fehlt das erste Anführungszeichen oder die öffnende eckige Klammer. Bei Verwendung eines Makros ist der aufgelöste Text falsch, weil er nicht in < > oder " " eingefaßt ist.

Bad filename format in include statement *MAKE-Fehler*

Ungültiges Dateinamensformat in der **include**-Anweisung

Zusammen mit **include** angegebene Dateinamen müssen in Anführungszeichen oder spitzen Klammern stehen. Das führende Anführungszeichen bzw. die spitze Klammer fehlt.

Bad file name format in line directive *Compiler-Fehler*

Ungültiger Dateiname in **#line**-Anweisung

Dateinamen für **#line** müssen entweder mit Anführungszeichen ("filename.h") oder mit spitzen Klammern (<filename.h>) eingefaßt sein. Dem angegebenen Dateinamen fehlt das erste Anführungszeichen oder die öffnende eckige Klammer. Bei Verwendung eines Makros ist der aufgelöste Text falsch, weil er nicht in " " eingefaßt ist.

Bad GCD type in GRPDEF, extended dictionary aborted *Bibliotheken-Warnung*

Unzulässiger GCD-Typ in GRPDEF, erweitertes Verzeichnis wird nicht angelegt

Bad GRPDEF type encountered, extended dictionary aborted *Bibliotheken-Warnung*

Unzulässiger GRPDEF-Typ, erweitertes Verzeichnis wird nicht angelegt

Die Bibliotheksverwaltung hat während der Erzeugung eines erweiterten Verzeichnisses einen unzulässigen Einsprung in ein Gruppendefinitions-Record (GRPDEF) eines Objektmoduls entdeckt. Die einzigen von der Bibliotheksverwaltung und dem Linker unterstützten GRPDEF-Recordtypen sind Segment-Indextypen. Es ist möglich, daß Objektmodule, die von an-

deren Zusatzprogrammen als denen von Borland erstellt werden, GRPDEF-Records von anderem Typ erzeugen. Diese Warnung wird auch erzeugt, wenn das Objektmodul zerstört worden ist.

Bad header in input LIB *Bibliotheken-Fehler*

Ungültiger Kopf in der Eingabebibliothek
Die Bibliotheksverwaltung hat beim Hinzufügen von Objektmodulen zu einer bestehenden Bibliothek festgestellt, daß der Kopf dieser Bibliothek ungültig ist. Bauen Sie die Bibliothek erneut auf.

Bad ifdef directive syntax *Compiler-Fehler*

Ungültige **#ifdef**-Direktive
Die **#ifdef**-Direktive darf nur den Namen genau eines Bezeichners enthalten und nichts anderes.

Bad LF_POINTER in module *Modulname* *Linker-Fehler*

Ungültiger LF_POINTER in Modul *Modulname*
Dieser Fehler wird typischerweise durch ungültige Debug-Informationen in der OBJ-Datei verursacht. Sie sollten Borland verständigen.

Bad macro output translator *MAKE-Fehler*

Ungültiger Ausgabe-Übersetzer im Makro
Ungültige Syntax für die Ersetzung in Makros, z.B.:

```
$(MODEL:=s)   oder   $(MODEL:)   oder   $(MODEL:s)
```

Bad object file *Dateiname* **near file offset** *Offset* *Linker-Fehler*

Ungültige Objektdatei *Dateiname* nahe Dateioffset *Offset*
Der Linker hat eine ungültige OBJ-Datei gefunden. Dieses wird üblicherweise durch einen Übersetzerfehler ausgelöst.

Bad object file record in library file *Dateiname* **in module** *Modul* **near module file offset 0xxxxxxxxx** *Linker-Fehler*

Ungültiger Objektdateiabschnitt im Modul *Modul* der Bibliotheksdatei *Dateiname* in der Nähe des Moduldateioffset 0xxxxxxxxx

Bad object file record in module *Modul* **near module file offset 0xxxxxxxxx** *Linker-Fehler*

Ungültiger Objektdateiabschnitt im Modul *Modul Dateiname* in der Nähe des Moduldateioffset 0xxxxxxxxx
Es wurde eine fehlerhafte Objektdatei festgestellt. Dieser Fehler wird im allgemeinen durch die Angabe einer Quelltext- oder Objektdatei verursacht, die nicht vollständig erstellt werden konnte. Er kann auftreten, wenn während der Compilierung gebootet wurde oder ein Compiler nach dem Drücken von [*Strg*] [*Break*] seine Ausgabe-Objektdatei nicht gelöscht hat.

Bad OMF record type *Typ* **encountered in module** *Modul* *Bibliotheken-Fehler*

Unzulässiger OMF-Record-Typ *Typ* in Modul *Modul*
Die Bibliotheksverwaltung hat einen unzulässigen Objekt-Modul-Format-Record (OMF) beim Lesen des Objekt-Moduls entdeckt. Da die Bibliotheksverwaltung die Header-Records des Moduls bereits gelesen und verifiziert hat, ist das Objekt-Modul wahrscheinlich zerstört. Erzeugen Sie es erneut.

Bad syntax for pure function definition *Compiler-Fehler*

Falsche Syntax für die Deklaration einer abstrakten Funktion
Abstrakte Funktionen kennzeichnet man durch ein **= 0** am Ende der Funktionsdeklaration. Sie haben etwas ähnliches geschrieben, aber nicht genau das gleiche.

Bad undef directive syntax *Compiler-Fehler*

Ungültige **#undef**-Direktive
Auf die #**undef**-Direktive muß der Name eines Bezeichners folgen, andere Angaben sind nicht zulässig.

Bad undef statement syntax *MAKE-Fehler*

Unzulässige Anwendung von **!undef**
Eine **!undef**-Anweisung darf ausschließlich einen einzelnen Bezeichner als Rumpf der Anweisung enthalten.

Bad version number in parameter block *Linker-Fehler*

Falsche Versionsnummer im Parameterblock

Dieser Fehler deutet auf eine interne Inkonsistenz in der IDE hin. Tritt dieser Fehler auf, verlassen Sie die IDE, und starten Sie sie neu. Dieser Fehler tritt in der Einzelplatz-Version nicht auf.

Base class *Klasse* contains dynamically dispatchable functions *Compiler-Fehler*

Basisklasse *Klasse* enthält dynamisch zugeordnete Funktionen

Im Augenblick unterstützen dynamische Zuweisungstabellen die Mehrfachvererbung nicht. Dieser Fehler tritt auf, weil eine Klasse, die eine DDVT-Funktion enthält, versucht hat, DDVT-Funktionen von mehreren Klassen zu erben.

Base class *Klasse* is inaccessible because also in *Klasse* *Compiler-Warnung*

Auf die Basisklasse *Klasse* kann nicht zugegriffen werden, weil sie sich auch in *Klasse* befindet

Die Verwendung einer Klasse sowohl als direkte als auch als indirekte Basisklasse ist nicht zulässig, da die Elemente automatisch mehrdeutig werden. Definieren Sie die Klasse an beiden Positionen als virtuell.

Base class *Klasse* is included more than once *Compiler-Fehler*

Basisklasse *Klasse* mehr als einmal vorhanden

Eine C++ Klasse kann von beliebig vielen Basisklassen abgeleitet werden. Sie kann jedoch nur einmal direkt von einer gegebenen Klasse abgeleitet werden.

Base class *Klasse* is initialized more than once *Compiler-Fehler*

Basisklasse *Klasse* mehr als einmal initialisiert

Im Konstruktor einer C++ Klasse enthält die Liste der Initialisierungen nach dem Konstruktorkopf mehr als einmal die Basisklasse *Klasse*.

Base initialization without a class name is now obsolete *Compiler-Fehler*

Initialisierung von Basisklassen ohne Klassennamen ist veraltet

Frühere Versionen von C++ initialisierten eine Basisklasse, indem auf den Kopf des Konstruktors lediglich die Parameterliste des Konstruktors der Basisklasse folgte. Jetzt sollte der Name der Basisklasse mit eingebunden werden. Dadurch wird der Quelltext lesbarer, was bei mehrfachen Basisklassen auch notwendig wird.

Früher:

```
derived::derived(int i) : (i,10) { ... }
```

Heute:

```
derived::derived(int i) : base(i,10) { ... }
```

Bit field cannot be static *Compiler-Fehler*

Bitfeld darf nicht statisch sein

Nur Datenelemente einer gewöhnlichen C++ Klasse können **static** deklariert werden, Bitfelder nicht.

Bit field too large *Compiler-Fehler*

Bitfeld zu groß

Dieser Fehler tritt auf, wenn ein Bitfeld größer als 32 Bits ist.

Bit fields must be signed or unsigned int *Compiler-Fehler*

Bitfelder müssen vom Typ signed oder unsigned **int** sein

In ANSI C dürfen Bitfelder nur vom Typ signed oder unsigned **int** sein (beispielsweise nicht vom Typ **char** oder **long**).

Bit fields must be signed or unsigned int *Compiler-Warnung*

Bitfelder müssen vom Typ signed oder unsigned **int** sein

In ANSI C können Bitfelder nicht vom Typ **char** oder unsigned **char** sein. Wenn nicht im strengen ANSI-Modus compiliert wird, läßt der Compiler diese Konstrukte zu, gibt jedoch Warnungen aus.

Bit fields must contain at least one bit *Compiler-Fehler*

Bitfelder müssen mindestens ein Bit enthalten

Ein benanntes Bitfeld darf nicht 0 Bit (oder weniger als 0) enthalten. Ein nicht benanntes Bitfeld darf 0 Bit enthalten, um das folgende Bitfeld an einer Byte-Grenze (oder Wortgrenze, wenn Sie die Wortausrichtung ausgewählt haben) auszurichten.

Bit fields must have integral type *Compiler-Fehler*

Bitfelder müssen Integertypen sein

In C++ müssen Bitfelder von einem ganzzahligen Typ sein; dazu gehören auch Aufzählungstypen **enum**).

Body has already been defined for function *Funktion* *Compiler-Fehler*

Funktionsrumpf für diese Funktion schon definiert

Für die Funktion *Funktion* dieses Namens und Typs existiert schon ein Funktionsrumpf. Ein Funktionsrumpf kann nur einmal definiert werden.

Both return and return of a value used *Compiler-Warnung*

return ohne Wertrückgabe und **return** mit Wertrückgabe benutzt

Die aktuelle Funktion hat **return**-Anweisungen mit und ohne Wert. Das ist in C zulässig, ergibt aber meistens einen Fehler. Möglicherweise wurde eine **return**-Anweisung am Ende der Funktion vergessen.

Call of nonfunction *Compiler-Fehler*

Angegebener Bezeichner ist kein Funktionsname

Bei dem angegebenen Bezeichner handelt es sich nicht um eine Funktion. Sie haben wahrscheinlich die Funktion falsch deklariert oder sich beim Funktionsnamen vertippt.

Call to function *Funktion* **with no prototype** *Compiler-Warnung*

Aufruf der Funktion *Funktion* ohne vorherige Definition eines Prototyps

Diese Warnung erhalten Sie nach Aktivierung von »Prototypes required«-Warnungen, wenn Sie die Funktion *Funktion* aufrufen, ohne zuvor einen Prototyp für diese Funktion angegeben zu haben.

Call to undefined function *Funktion* *Compiler-Fehler*

Aufruf einer nicht definierten Funktion *Funktion*

In Ihrer Quelldatei haben Sie eine Funktion deklariert, die einen anderen Typ als **void** (oder **int** in C) zurückgibt, aber der Compiler hat eine **return**-Anweisung mit keinem Rückgabewert gefunden. Typischerweise ist folgender Fehler gemeint: **int**-Funktionen müssen in C keinen Wert zurückliefern. In älteren C-Versionen war der Typ **void**, der anzeigt, daß kein Wert zurückgeliefert werden soll, nicht implementiert.

virtual can only be used with member funktons *Compiler-Fehler*

virtual kann nur bei einer Elementfunktion benutzt werden

Ein Datenelement wurde mit dem **virtual**-Spezifizierer deklariert. Nur Elementfunktionen könne **virtual** deklariert werden. So zum Beispiel:

```
class myclass
{
public:
   virtual int a;      // Fehler
};
```

Cannot access an inactive scope *Compiler-Fehler*

Kein Zugriff auf einen nicht-aktiven Gültigkeitsbereich möglich

Sie haben versucht, eine zu einer nicht aktiven Funktion lokale Variable auszuwerten oder zu untersuchen.

Cannot add or subtract relocatable symbols *Compiler-Fehler*

Relozierbare Symbole können nicht addiert oder subtrahiert werden

Die einzige arithmetische Operation, die für ein relozierbares Symbol in einem Assembler-Operanden ausgeführt werden kann, ist die Addition oder Subtraktion einer Konstanten. Variablen, Prozeduren, Funktionen und Labels sind relozierbare Symbole. Wenn *Var* eine Variable und *Const* eine Konstante ist, dann sind die Befehle

```
MOV AX, Const+Const
```

und

```
MOV AX, Var+Const
```

gültig

```
MOV AX, Var+Var
```

dagegen nicht.

Cannot allocate a reference *Compiler-Fehler*

Für eine Referenz kann kein Speicherplatz reserviert werden
Sie wollen eine Referenz mittels des Operators **new** erstellen. Dies ist nicht erlaubt, weil Referenzen keine Objekte sind und nicht durch **new** erzeugt werden können.

Bezeichner **cannot be declared in an anonymous union** *Compiler-Fehler*

Bezeichner kann nicht in einer anonymen Variante deklariert werden
Der Compiler hat eine Deklaration einer Elementfunktion oder eines statischen Elements in einer anonymen Variante gefunden. Solche Varianten können nur Datenelemente enthalten.

Funktion1 **cannot be distinguished from** *Funktion2* *Compiler-Fehler*

Funktion1 nicht von *Funktion2* unterscheidbar
Die Parameterlisten in den Deklarationen dieser Funktionen sind nicht unterscheidbar. Ändern Sie die Reihenfolge oder den Typ eines Parameters einer Deklaration.

Cannot call main from within the program *Compiler-Fehler*

Der Aufruf von **main** innerhalb eines Programms ist nicht möglich
C++ erlaubt die rekursiven Aufrufe der Funktion **main** nicht.

Cannot call near class member function with a pointer of type *Typ* *Compiler-Fehler*

Elementfunktion einer **near**-Klasse kann nicht über Zeiger vom Typ *Typ* aufgerufen werden
Elementfunktionen von **near**-Klassen können nicht mit Elementzeigern vom Typ **far** oder **huge** aufgerufen werden. (Zur Erinnerung: Klassen sind in den Speichermodellen *Tiny*, *Small* und *Medium* per Voreinstellung vom Typ **near**.) Dies trifft auch auf Aufrufe mit Zeigern auf Elementfunktionen zu. Ändern Sie entweder den Zeiger auf den Typ **near** um, oder deklarieren Sie die Klasse als **far**.

Cannot cast from *Typ1* **to** *Typ2* *Compiler-Fehler*

Typenumwandlung von *Typ1* nach *Typ2* nicht zulässig
Eine Umwandlung von *Typ1* in *Typ2* ist nicht zulässig. In C kann ein Zeiger in einen anderen Zeiger oder einen Integertyp umgewandelt werden. Ein Integertyp kann in jeden Integer-, Gleitkomma- oder Zeigertyp umgewandelt werden, ein Gleitkomma- in einen Gleitkomma- oder Integertyp. Strukturen und Arrays können nicht ineinander oder in andere Typen umgewandelt werden. Der Datentyp **void** kann ebensowenig umgewandelt werden.

In C++ werden benutzerdefinierte Umwandlungsvorschriften und Konstruktoren geprüft. Sind keine benutzerdefinierten Regeln vorhanden, gelten die oben genannten Regeln (außer für Zeiger auf Klassenelemente). Bei Integertypen darf nur die Konstante NULL in den Zeiger auf ein Element umgewandelt werden. Ein Elementzeiger kann in einen Integertyp oder in einen ähnlichen Elementzeiger umgewandelt werden. Ein ähnlicher Zeiger ist ein Zeiger auf ein Datenelement, falls der erste Zeiger dorthin zeigt, oder auf eine Elementfunktion, falls der erste Zeiger auf eine solche zeigt. Die qualifizierende Klasse des umzuwandelnden Typs muß dieselbe oder eine Basisklasse des ursprünglichen Typs sein.

Cannot convert *Typ1* **to** *Typ2* *Compiler-Fehler*

Kann *Typ1* nicht in *Typ2* umwandeln
Eine Zuweisung, eine Initialisierung oder ein Ausdruck erfordern die besagte Typumwandlung; diese ist jedoch nicht erlaubt.

Cannot create instance of abstract class *Klasse* *Compiler-Fehler*

Kann für abstrakte Klasse *Klasse* keine Variable erzeugen
Abstrakte Klassen (solche mit abstrakten Funktionen) können nicht direkt verwendet werden, d.h es kann keine Instanz erzeugt werden. Es kann von ihnen nur abgeleitet werden.

Cannot define a pointer or reference to a reference *Compiler-Fehler*

Ein Zeiger oder eine Referenz auf eine Referenz ist nicht zulässig
Ein Zeiger oder eine Referenz auf eine Referenz ist nicht zulässig.

Cannot find *Klasse*::*Klasse* **(***Klasse***&) to copy a vector** *Compiler-Fehler*

Kann Vektor nicht kopieren, Referenz auf *Klasse* fehlt
Enthält eine C++ Klasse *Klasse1* einen Vektor (Array) aus Klasse *Klasse2* und Sie wollen ein Objekt des Typs *Klasse1* aus einem anderen Objekt dieser Klasse erstellen, müssen Sie diesen Konstruktor verwenden:

```
Klasse2::Klasse2(Klasse2&)
```

Dadurch können die Elemente des Vektors kopiert werden. Der Konstruktor, Kopierkonstruktor genannt, braucht nur einen Parameter als Referenz zu seiner Klasse.

Der Kopierkonstruktor wird üblicherweise vom Compiler generiert. Wenn Sie aber einen Konstruktor für die Klasse *Klasse2* definiert haben mit einem Parameter des Typs *Klasse2&* und mit zusätzlichen Parametern mit Voreinstellung, dann kann kein Kopierkonstruktor existieren oder vom Compiler erzeugt werden.

Der Grund dafür ist, daß die folgenden Deklarationen nicht unterschieden werden können:

```
Klasse2::Klasse2(Klasse2&)
Klasse2::Klasse2(Klasse2&, int = 1)
```

Sie müssen den Konstruktor so definieren, daß nicht alle Parameter Vorgabewerte haben. Dann können Sie einen Kopierkonstruktor definieren oder vom Compiler erzeugen lassen.

Cannot find *Klasse***::operator** *=* **(***Klasse&***) to copy a vector** *Compiler-Fehler*

Zum Kopieren eines Array kann kein Operator *Klasse*::operator=(*Klasse&*) gefunden werden
Enthält eine C++ Klasse *Klasse1* einen Vektor (Array) aus Klasse *Klasse2*, und Sie wollen ein Objekt des Typs *Klasse1* kopieren, müssen Sie diesen Zuweisungsoperator verwenden:

```
Klasse2::operator=(Klasse2&)
```

Dadurch können die Elemente des Vektors kopiert werden. Normalerweise wird dieser Operator vom Compiler bereitgestellt.

Dies ist jedoch nicht der Fall, wenn Sie einen **operator=** für die Klasse *Klasse2* definiert haben, der keinen Parameter des Typs *Klasse2&* übernimmt. In diesem Fall müssen Sie selbst einen solchen Operator zur Verfügung stellen.

Cannot find default constructor to initialize array element of type *Klasse* *Compiler-Fehler*

Zur Initialisierung eines Array-Elements vom Typ *Klasse* kein Standardkonstruktor vorhanden
Bei der Deklaration eines Arrays über eine Klasse, die Konstruktoren besitzt, müssen Sie entweder jedes einzelne Element im Array explizit initialisieren, oder die Klasse muß über einen Standardkonstruktor verfügen. (Er wird benutzt, um die Array-Elemente zu initialisieren, die keinen expliziten Initialisierer besitzen.) Solange Sie für die Klasse keinen Konstruktor definiert haben, stellt der Compiler einen Standardkonstruktor bereit.

Cannot find default constructor to initialize base class *Klasse* *Compiler-Fehler*

Zur Initialisierung der Basisklasse *Klasse* muß ein Standardkonstruktor existieren

Wenn in C++ eine abgeleitete Klasse *Klasse2* erstellt wird, muß eine Basisklasse *Klasse1* schon existieren. Falls der Konstruktor für *Klasse2* keinen Konstruktor von *Klasse1* (im Kopf von *Klasse2*) angibt, muß es einen Konstruktor *Klasse1*::*Klasse1*() für die Basisklasse geben. Dieser parameterlose Konstruktor heißt Standardkonstruktor. Der Standardkonstruktor wird üblicherweise vom Compiler bereitgestellt, außer es existiert schon ein Konstruktor für die Klasse *Klasse1*. In diesem Fall müssen Sie selbst einen Konstruktor bereitstellen.

Cannot find default constructor to initialize member *Bezeichner* *Compiler-Fehler*

Zur Initialisierung des Elements *Bezeichner* muß ein Standardkonstruktor existieren

Wenn eine C++ Klasse *Klasse1* ein Element der Klasse *Klasse2* enthält und Sie ein Objekt der *Klasse1* nicht aus einem anderen Objekt der *Klasse1* erstellen wollen, muß ein Konstukrtor *Klasse2*::*Klasse2*() existieren, damit das Element erzeugt werden kann. Dieser Konstruktor ohne Parameter wird Standardkonstruktor genannt. Der Compiler liefert automatisch einen Standardkonstruktor, außer wenn Sie bereits einen Konstruktor für die *Klasse2* definiert haben. In diesem Fall stellt der Compiler den Standardkonstruktor nicht zur Verfügung; er muß von Ihnen geliefert werden.

Cannot find MAKE.EXE *MAKE-Fehler*

MAKE.EXE kann nicht gefunden werden

Das Kommandozeilen-Hilfsprogramm MAKE kann nicht gefunden werden. Vergewissern Sie sich, daß sich MAKE im aktuellen Verzeichnis oder in einem Verzeichnis aus Ihrem Zugriffspfad befindet.

Cannot generate COM file: data below initial CS:IP defined *Linker-Fehler*

Nicht als COM-Datei linkbar: Daten unterhalb CS:IP

Es wurde versucht, Daten oder Code unterhalb der Startadresse (0x0100) einer COM-Datei zu generieren. Überprüfen Sie, ob die Startadresse mit der Anweisung ORG 100H gesetzt ist. Dieser Fehler tritt normalerweise in einem Hochsprachenprogramm nicht auf; sollte dies doch der Fall sein, müssen Sie sicherstellen, daß das richtige Startup-Objectmodul (C0x) gelinkt wird.

Cannot generate COM file: invalid initial entry point address *Linker-Fehler*

Nicht als COM-Datei linkbar: ungültige Adresse für den ursprünglichen Einsprungspunkt

Sie haben zwar die Option **/Tdc** oder **/t** verwendet, aber die Startadresse des Programms ist nicht 100H, wie es für COM-Dateien erforderlich ist.

Cannot generate COM file: program exceeds 64K *Linker-Fehler*

Nicht als COM-Datei linkbar: Programm größer als 64KByte

Sie benutzten die **/Tdc**- oder die **/t**-Option, aber das Programm überschreitet die zulässige Größe für COM-Dateien.

Cannot generate COM-File: segment-relocatable items present *Linker-Fehler*

Nicht als COM-Datei linkbar: Programm enthält zu relozierende Einträge

Sie haben zwar die Option **/Tdc** oder **/t** verwendet, aber das Programm enthält segmentbezogene Fixups, die in COM-Dateien nicht zugelassen sind.

Cannot generate COM-File: stack segment present *Linker-Fehler*

Nicht als COM-Datei linkbar: Stack-Segment vorhanden

Sie benutzten die **/Tdc-** oder die **/t**-Option, aber im Programm wird ein Stack-Segment deklariert, das in COM-Dateien nicht zulässig ist.

Cannot generate *Funktion* **from template function** *Schablone* *Compiler-Fehler*

Funktion kann nicht aus der Template-Funktion *Schablone* generiert werden

Es liegt ein Aufruf einer Template-Funktion vor; aus der Funktion *Schablone* kann aber keine entsprechende Template-Funktion erzeugt werden.

Cannot have a non-inline function in a local class *Compiler-Fehler*

Nicht **inline**-Funktion in lokaler Klasse nicht möglich

Cannot have a static data member in a local class *Compiler-Fehler*

static Datenelemente in lokaler Klasse nicht möglich

Alle zu einer Funktion lokal deklarierten Elemente einer Klasse müssen vollständig in der Definition der Klasse definiert sein. Dies bedeutet, daß lokale Klassen keine statischen Datenelemente enthalten können und die Funktionsrümpfe aller Elementfunktionen innerhalb der Definition der Klasse definiert sein müssen.

Cannot have multiple paths for implicit rule *MAKE-Fehler*

Für Erweiterungen in einer impliziten Regel darf es nicht mehrere Pfade geben

Für jede Erweiterung in einer impliziten Regel darf es nur eine Pfadangabe geben. Mehrere Suchpfade sind nur für Abhängigkeiten in einer expliziten Regel erlaubt:

```
{path1;path2}.c.obj:   // Ungültig
{path}.c.obj // Gültig
```

Cannot have path list for target *MAKE-Fehler*

Pfadliste für Ziel nicht zulässig

Sie können nur für Abhängigkeiten in einer expliziten Regel eine Pfadliste angeben:

```
{path1;path2}prog.exe: prog.obj  // Ungültig
prog.exe: {path1;path2}prog.obj  // Gültig
```

Cannot initialize a class member here *Compiler-Fehler*

Kann Feld nicht initialisieren

Einzelne Felder einer Struktur, Union oder Klasse in C++ können nicht initialisiert werden. Eine Struktur oder Union kann als Ganzes mit Initialisierungswerten in geschweiften Klammern initialisiert werden. Eine C++ Klasse kann nur durch einen Konstruktor initialisiert werden.

Cannot initialize *Typ1* **with** *Typ2* *Compiler-Fehler*

Typ1 kann nicht mit *Typ2* initialisert werden

Sie haben versucht, ein Objekt vom Typ *Typ1* mit einem Wert vom Typ *Typ2* zu initialisieren. Das ist jedoch nicht erlaubt. Für Initialisierungen gelten grundsätzlich die gleichen Regeln wie für Zuweisungen.

Cannot modify a const object *Compiler-Fehler*

Ein **const**-Objekt kann nicht verändert werden

Diese Meldung signalisiert, daß für ein **const** deklariertes Objekt eine unzulässige Operation durchgeführt wurde, wie z.B. eine Zuweisung an das Objekt.

Cannot overload main *Compiler-Fehler*

main kann nicht überladen werden

main ist die einzige Funktion, die nicht überladen werden kann.

Funktion **cannot return a value** *Compiler-Fehler*

Funktion kann keinen Wert zurückliefern

Eine Funktion mit einem Rückgabewert vom Typ **void** enthält eine **return**-Anweisung, die z.B. ein **int** zurückliefert, dies ist nicht möglich.

Bezeichner **cannot start a parameter declaration** *Compiler-Fehler*

Bezeichner kann keine Parameterdeklaration einleiten

Ein nicht definierter Bezeichner wurde am Anfang eines Arguments in einer Funktionsdeklaration entdeckt. Oft ist der Typenname falsch geschrieben oder die Typendeklaration fehlt. Der Grund hierfür ist normalerweise die fehlende Einbindung der entsprechenden Header-Datei.

Bezeichner **cannot start an argument declaration** *Compiler-Fehler*

Bezeichner kann keine Argument-Deklaration einleiten

In einer Funktionsdeklaration wurde ein undefinierter Bezeichner am Beginn eines Arguments gefunden.

Cannot take address of main *Compiler-Fehler*

Kann auf Adresse von **main** nicht zugreifen

In C++ ist der Zugriff auf die Adresse des Hauptprogramms **main** nicht zulässig.

Cannot throw *Typ* **- ambiguous base class** *Basis* *Compiler-Fehler*

Mehrdeutige Basisklasse *Basis*

Eine Klasse kann nicht mehr als eine Kopie einer (nicht-virtuellen) Basisklasse enthalten.

Cannot write GRPDEF list, extended dictionary aborted *Bibliotheken-Fehler*

GRPDEF-Liste kann nicht geschrieben werden, erweitertes Verzeichnis wird nicht angelegt

Die Bibliotheksverwaltung kann das erweiterte Verzeichnis nicht an das hintere Ende der Bibliotheksdatei schreiben. Es ist möglicherweise nicht mehr genügend Platz auf der Platte/Diskette.

Can't grow LE/LIDATA record buffer *Bibliotheken-Fehler*

LE/LIDATA-Puffer kann nicht vergrößert werden

Kommandozeilenfehler. Vergleiche die Meldung »out of memory LE/LIDATA record from object module«.

Case bypasses initialization of a local variable *Compiler-Fehler*

Case umgeht die Initialisierung einer lokalen Variablen

Es ist in C++ nicht zulässig, die Initialisierung einer lokalen Variablen zu umgehen. Dieser Fehler deutet auf ein **case**-Label hin, welches die Steuerung hinter diese lokale Variable transferiert.

Case outside of switch *Compiler-Fehler*

Case-Label außerhalb von **switch**

Der Compiler hat eine **case**-Anweisung angetroffen, die außerhalb einer **switch**-Anweisung steht. Hier handelt es sich normalerweise um eine falsch plazierte geschweifte Klammer.

Case statement missing : *Compiler-Fehler*

Case-Anweisung ohne Doppelpunkt

Auf das reservierte Wort **case** muß eine einzelne Konstante und darauf ein Doppelpunkt folgen. In der **case**-Anweisung fehlt entweder der Doppelpunkt im Ausdruck, oder es steht ein zusätzliches Zeichen vor dem Doppelpunkt.

catch expected *Compiler-Fehler*

catch erwartet

In C++ muß einem **try**-Block mindestens ein **catch**-Block folgen.

Character constant must be one or two characters long *Compiler-Fehler*

Zeichenkonstante zu lang

Zeichenkonstanten dürfen nur aus einem oder zwei Zeichen bestehen.

Circular dependency exists in makefile *MAKE-Fehler*

In der MAKE-Datei existiert eine zirkuläre Abhängigkeit

Die MAKE-Datei zeigt an, daß eine Datei aktualisiert werden muß, bevor sie angelegt werden kann. Ein Beispiel für explizite Regeln:

```
filea: fileb
fileb: filec
filec: filea
```

Es wird festgelegt, daß file*a* abhängig ist von file*b*, die wieder von file*c* abhängt. Schließlich hängt file*c* von file*a* ab. Diese Festlegung ist unzulässig, da eine Datei weder direkt noch indirekt von sich selbst abhängen darf.

Class *Klasse* **may not contain pure functions** *Compiler-Fehler*

Klasse *Klasse* darf keine abstrakte Funktion enthalten

Die zu deklarierende Klasse kann nicht abstrakt sein, und darum kann sie auch keine abstrakten Funktionen enthalten.

Class member *Element* declared outside its class *Compiler-Fehler*

Klassen-Element *Element* außerhalb seiner Klasse deklariert

Elementfunktionen einer C++ Klasse können nur in der Klassendeklaration deklariert werden. Sie können nicht wie andere Funktionen mehrfach oder an anderen Stellen deklariert werden.

Code has no effect *Compiler-Warnung*

Erzeugter Code ist wirkungslos

Diese Warnung erhalten Sie, wenn der Compiler auf eine Anweisung mit Operatoren trifft, die keine Auswirkung hat. Beispielsweise hat folgende Anweisung.

```
a + b;
```

keine Auswirkung auf eine der beiden Variablen. Die Operation ist nicht erforderlich und weist wahrscheinlich auf einen Fehler hin.

Colon expected *MAKE-Fehler*

Doppelpunkt fehlt

Sie haben den Doppelpunkt am Ende der impliziten Regel vergessen:

```
.c.obj:   // Richtig
.c.obj // Falsch
```

Command arguments too long *MAKE-Fehler*

Kommandozeilenparameter zu lang

Die Länge der Parameter eines Kommandos hat die von DOS zugelassene Grenze von 511 Zeichen überschritten.

Command syntax error *MAKE-Fehler*

Syntaxfehler

Diese Fehlermeldung kann eine Reihe von Ursachen haben:

- Die erste Regel in der MAKE-Datei enthält am Anfang Whitespace-Zeichen.
- Eine implizite Regel wurde nicht als *.ext.ext:* definiert.
- In einer expliziten Regel steht vor dem Doppelpunkt kein Name.
- In einer Makrodefinition fehlt vor dem Gleichheitszeichen der Name.

Command too long *MAKE-Fehler*

Befehl zu lang

Der Befehl hat mehr als 512 Zeichen. Sie sollten eine Response-Datei verwenden.

Common segment exceeds 64K *Linker-Fehler*

Gemeinsames Segment ist größer als 64 KByte

Das Programm hat mehr als 64 KByte nicht initialisierter **near** Daten. Versuchen Sie, einige nicht initialisierte Daten als **far** zu deklarieren.

Compiler could not generate copy constructor for class *Klasse* *Compiler-Fehler*

Compiler konnte den Kopierkonstruktor für die Klasse *Klasse* nicht erzeugen.

Der Compiler konnte den erforderlichen Kopierkonstruktor nicht den Sprachregeln gemäß erstellen.

Compiler could not generate operator= for class *Klasse* *Compiler-Fehler*

Compiler konnte **operator=** für die Klasse *Klasse* nicht erzeugen

Der Compiler konnte den erforderlichen Zuweisungsoperator nicht den Sprachregeln gemäß erzeugen.

Compiler table limit exceeded *Compiler-Fehler*

Überlauf in einer Compiler-Tabelle

Eine der internen Tabellen des Compilers ist übergelaufen. Dies bedeutet in der Regel, daß das compilierte Modul zu viele Funktionskörper enthält. Diese Beschränkung wird auch dadurch nicht aufgehoben, daß Sie dem Compiler mehr Speicher zur Verfügung stellen. Sie sollten die zu compilierende Datei vereinfachen.

Compound statement missing } *Compiler-Fehler*

Blockanweisung: abschließendes **}** fehlt
Der Compiler hat das Ende der Quelltextdatei erreicht und keine rechte, geschweifte Klammer gefunden; d.h. es besteht ein Klammerungsfehler.

Condition is always false *Compiler-Warnung*
Condition is always true *Compiler-Warnung*

Bedingung ist immer *false*/*true*
Der Compiler fand eine Bedingung, die nach ihrer Auswertung immer *true*/*false* zurückliefert, z.B.:

```
void proc(unsigned x) {
   if (x >= 0) {      /* immer true */
   }
}
```

Conflicting type modifiers *Compiler-Fehler*

Angegebene Modifizierer nicht miteinander vereinbar
Es ist eine Deklaration vorhanden, die z.B. das Schlüsselwort **far** und **near** für denselben Zeiger enthält. Es darf aber nur ein Adreß-Modifizierer für einen Zeiger und nur ein Sprach-Modifizierer **cdecl**, **pascal** oder **interrupt**) für eine Funktion existieren.

Symbol **conflicts with module** *Modulname* **in module** *Modulname* *Linker-Warnung*

Symbol ist unverträglich mit Modul *Modulname* im Modul *Modulname*
Diese Warnung signalisiert eine Inkonsistenz in der Definition von Symbol. Das bedeutet, daß TLINK eine virtuelle Funktion und eine allgemeine Definition gefunden hat, die denselben Namen verwenden.

Constant expression required *Compiler-Fehler*

Angegebener Ausdruck ist keine Konstante
Arrays müssen mit einer konstanten Größe deklariert werden. Dieser Fehler wird häufig durch einen Tippfehler in einer **#define**-Konstante verursacht.

Constant is long *Compiler-Warnung*

Konstante wird als **long** gespeichert
Der Compiler ist entweder auf eine Dezimalkonstante, die größer als 32767 ist, oder auf eine Oktalkonstante, die größer als (dezimal) 65535 ist, gestoßen. Bei diesen Konstanten fehlt der explizite Zusatz von **l** oder **L**. Die Konstante wird als Konstante vom Typ **long** behandelt.

Constant member *Element* **in class without constructors** *Compiler-Fehler*

Element in Klasse ohne Konstruktor
Eine Klasse, die konstante Elemente enthält, muß mindestens einen vom Benutzer definierten Konstruktor aufweisen. Andernfalls gäbe es keine Möglichkeit, solche Elemente zu initialisieren.

Constant member *Element* **is not initialized** *Compiler-Warnung*

Konstantes Element *Element* nicht initialisiert
Diese C++ Klasse enthält ein konstantes Element *Element*, das nicht initialisiert wurde. Beachten Sie, daß konstante Elemente nur initialisiert werden dürfen; ihnen kann nichts zugewiesen werden.

Constant out of range in comparison *Compiler-Warnung*

Vergleich verschiedener Teilbereiche
Ihre Quelltextdatei enthält einen Vergleich, der eine Teilbereichskonstante betrifft, die außerhalb des Bereichs der anderen Konstante liegt. So ergibt z.B. der Vergleich eines **unsigned**-Typs mit -1 keinen Sinn. Um einen **unsigned**-Konstante,

die (dezimal) größer als 32767 ist, zu erhalten, sollten Sie entweder für **unsigned** eine Typumwandlung durchführen oder den Buchstaben **u** oder **U** an die Konstante anfügen.

Trotz dieser Meldung wird der Compiler Code generieren, um den Vergleich durchzuführen. Auch wenn der Code letztendlich immer dasselbe Resultat liefert (weil z.B. ein Ausdruck vom Typ **char** mit 4000 verglichen wird), wird der Code den Test doch immer wieder durchführen.

Constant variable *Variable* **must be initialized** *Compiler-Fehler*
Konstante Variable *Variable* nicht initialisiert
Dieses C++ Objekt ist als **const** deklariert, aber nicht initialisiert. Da ihm kein Wert zugewiesen werden kann, muß es bei der Deklaration initialisiert werden.

Constructor cannot be declared const or volatile *Compiler-Fehler*
Konstruktor kann nicht als **const** oder **volatile** deklariert werden
Ein Konstruktor wurde als **const** und/oder **volatile** deklariert. Das ist nicht zulässig.

Constructor cannot have a return type specification *Compiler-Fehler*
Rückgabetyp eines Konstruktors kann nicht spezifiziert werden
Konstruktoren in C++ haben einen impliziten Rückgabetyp, der vom Compiler benutzt wird. Sie dürfen selbst keinen Rückgabetyp oder -wert deklarieren.

Conversion may lose significant digits *Compiler-Warnung*
Konvertierung hat eventuell den Verlust signifikanter Ziffern zur Folge
Für einen Zuweisungsoperator oder ähnliches erfordert Ihre Quelltextdatei möglicherweise eine Konvertierung vom Typ **long** oder **unsigned long** nach **int** oder **unsigned int**. Da Variablen vom Typ **long** und **int** nicht dieselbe Größe haben, könnte diese Konvertierung das Verhalten eines Programms verändern.

Conversion of near pointer not allowed *Compiler-Fehler*
Konvertierung eines **near**-Zeigers nicht möglich
Wenn aktuell kein Programm läuft, kann in der Expression-Box ein **near**-Zeiger nicht in einen **far**-Zeiger umgewandelt werden. Dafür würde nämlich der aktuelle Wert des DS-Registers des Benutzerprogramms benötigt, der aber nicht existiert.

Conversion operator cannot have a return type specification *Compiler-Fehler*
Rückgabetyp eines Konvertierungsoperators kann nicht spezifiziert werden
Diese Typumwandlungsfunktion für ein C++ Element gibt an, daß ein abweichender Typ zurückgeliefert wird. Eine Deklaration für die Umwandlungsfunktion darf überhaupt keinen Rückgabetyp angeben.

Conversion to *Typ* **will fail for members of virtual base** *Klasse* *Compiler-Fehler*
Elemente der virtuellen Basisklasse *Klasse* können nicht in Typ *Typ* konvertiert werden
Diese Meldung tritt in Fällen auf, in denen ein Elementzeiger in einen anderen Elementzeigertyp umgewandelt wird, wenn die Klasse des Elementzeigers abstrakte Basisklassen enthält, und nur dann, wenn die Option **-Vv** oder die IDE-Option *Optionen/Projekt/C++ Optionen/C++ Kompatibilität/'Tiefe' virtuelle Basisklasse* verwendet wurde. Das heißt: Wenn der umzuwandelnde Elementzeiger zum Zeitpunkt der Umwandlung auf ein Element von *Klasse* zeigt, kann die Umwandlung nicht vollständig ausgeführt werden. Das Ergebnis der Umwandlung ist dann ein NULL-Zeiger.

Could not allocate memory for per module data *Bibliotheken-Fehler*
Nicht genügend Speicher für Modul-Daten
Die Bibliotheksverwaltung hat nicht mehr genügend Speicher zur Verfügung.

Could not create list file *Dateiname* *Bibliotheken-Fehler*
Listendatei *Dateiname* konnte nicht erzeugt werden
Die Bibliotheksverwaltung konnte für die Bibliothek keine Listendatei erzeugen. Dies kann an fehlendem Platten-/Disketten-Speicher liegen.

Could not find a match for *Argument(e)* — *Compiler-Fehler*

Für *Argumente* keine Funktion mit passenden Parametern gefunden

Es konnte keine C++ Funktion mit Parametern gefunden werden, die zu den angegebenen Argumenten passen.

Could not find *Dateiname* — *Compiler-Fehler*

Datei nicht gefunden

Der Compiler kann die in der Kommandozeile angegebene Datei nicht finden.

Could not get procedure address from DLL *Dateiname* — *Linker-Fehler*

Kann die Adresse der DLL-Prozedur nicht finden

Der Linker war nicht in der Lage, eine Prozedur aus einer bestimmten DLL zu finden. Vergewissern Sie sich, daß Sie die richtige DLL-Version benutzt haben.

Could not load DLL *Dateiname* — *Linker-Fehler*

Kann die DLL *Dateiname* nicht laden

Der Linker konnte die angegebene DLL nicht laden. Vergewissern Sie sich, daß die DLL in Ihrem Pfad ist.

Could not write output — *Bibliotheken-Fehler*

Keine Ausgabe möglich

Die Bibliotheksverwaltung kann die Ausgabedatei nicht anlegen

Couldn't alloc memory for per module data — *Bibliotheken-Fehler*

Keine Speicherplatzreservierung für Moduldaten möglich

Es ist nicht genügend Speicherkapazität für die Bibliotheksverwaltung vorhanden.

Dateiname **couldn't be created, original won't be changed** — *Bibliotheken-Warnung*

Dateiname konnte nicht angelegt werden, Originaldatei wurde nicht geändert

Es wurde versucht, ein Objekt *-Aktion) zu extrahieren. Die Bibliotheksverwaltung kann aber die Objektdatei nicht anlegen, in die das Modul extrahiert werden soll. Entweder existiert das Objekt bereits und kann nur gelesen werden, oder es ist kein Platz mehr vorhanden.

Couldn't get LE/LIDATA record buffer — *Bibliotheken-Fehler*

Konnte LE/LIDATA-Record-Puffer nicht bekommen

Kommandozeilenfehler - Vergleichen Sie mit der Meldung: »Out of memory reading LE/LIDATA record from object module«.

Debug info switch ignored for COM files — *Linker-Warnung*

Info-Schalter des Debuggers wurde für COM-Dateien ignoriert

Borland C++ kann keine Debug-Informationen für COM-Dateien generieren.

Debug information enabled, but no debug information found in OBJs — *Linker-Warnung*

Debugging-Informationen sind eingeschaltet, aber es wurden keine Debug-Informationen in den OBJ-Dateien gefunden

Alle Teile der Anwendung sind ohne die Debugging-Informationen compiliert worden, aber sie haben verlangt, daß sie beim Linken verwendet werden sollen.

Debug information in module *Modulname* **will be ignored** — *Linker-Warnung*

Debugging-Informationen im Modul *Modulname* werden ignoriert

Objekt-Dateien, die mit Debugging-Informationen compiliert werden, besitzen neuerdings einen Versions-Record. Die Versionsnummer von diesem Record ist höher als die momentan von TLINK unterstützte.

TLINK hat für das betreffende Modul keine Debugging-Informationen erzeugt.

Debugging information overflow; try fewer modules with debug info — *Linker-Fehler*

Überlauf der Debugging-Informationen. Verwenden Sie weniger Module mit Debugging-Informationen

Es sind zu viele Module mit Debugging-Informationen in den Link-Vorgang mit einbezogen. Compilieren Sie Ihr Programm neu, wobei Sie für weniger Module Debugging-Informationen vorsehen sollten.

Declaration does not specify a tag or an identifier *Compiler-Fehler*

Deklaration ohne Tag oder Bezeichner

Diese Deklaration deklariert gar nichts. Sie haben vielleicht eine Struktur oder eine Union ohne Benennung oder ohne Variablen deklariert. C++ verlangt, daß etwas deklariert wird.

Declaration is not allowed here *Compiler-Fehler*

Keine Deklaration erlaubt

Deklarationen dürfen nicht als Steueranweisungen in Ausdrücken wie **while**, **for**, **do**, **if** oder **switch** verwendet werden.

Declaration missing ; *Compiler-Fehler*

Es fehlt ein Strichpunkt in der Deklaration

Ihre Quelltextdatei enthält eine Deklaration ohne abschließenden Strichpunkt.

Deklaration syntax error *Compiler-Fehler*

Ungültige Syntax in einer Deklaration

Ihre Quelltextdatei enthält eine Deklaration, in der ein Symbol fehlt oder die überzälige Symbole aufweist.

Declaration terminated incorrectly *Compiler-Fehler*

Ende der Deklaration nicht korrekt

Nach der Deklaration steht ein überflüssiges oder falsches Zeichen, wie z.B. ein Semikolon nach einem Funktionsrumpf. Bei einer C++ Elementfunktion, die in einer Klasse zwischen Kopf und der ersten linken Klammer deklariert wird, taucht dieser Fehler ebenfalls auf.

Declaration was expected *Compiler-Fehler*

Deklaration erwartet

Hier wurde eine Deklaration erwartet, aber nicht gefunden. Dies ist meist mit einem fehlenden Begrenzer wie Komma, Strichpunkt, rechte Klammer usw. zu erklären.

Declare operator delete void*) or void*, size_t) *Compiler-Fehler*

Operator **delete** muß als **void***) oder **void***, size_t) deklariert werden

Korrekt deklarieren Sie **delete** mit einem einzelen **void***-Parameter oder einem zweiten zusätzlichen Parameter des Typs size_t. Die zweite Version hat Vorrang vor der ersten. Der globale Operator **delete** kann nur mit einem einzelnen Parameter deklariert werden.

Declare operator delete] void*) or void*, size_t) *Compiler-Fehler*

Operator **delete []** muß als (**void***) oder (**void***, size_t) deklariert werden

Deklarieren Sie den Operator **delete** nur mit einem **void*** Parameter oder mit **void*** und einem Parameter vom Typ size_t. Wenn Sie die zweite Möglichkeit nutzen wird sie der ersten vorgezogen. Der globale Operator **delete** kann nur in der Einparameter-Form benutzt werden.

Declare type *Typ* prior to use in prototype *Compiler-Warnung*

Der Typ *Typ* muß vor seiner Benutzung im Prototyp deklariert werden

Im Prototyp einer Funktion wird auf einen nicht vorher deklarierten Struktur-Typ Bezug genommen. Die beiden Deklarationen im Prototyp der Funktion und außerhalb sind deshalb nicht identisch, z.B. bei:

```
int func(struct s *ps);
struct s { /* ... */ };
```

Die Struktur **struct** *s* befindet sich nicht im Gültigkeitsbereich der Funktion *func*. Deshalb ist der Typ des Parameters *ps* ein Zeiger auf eine undefinierte Struktur **struct** *s* und kein Zeiger auf die später definierte Struktur **struct** *s*. Diese Warnung kann der Grund für nachfolgende seltsame Warnungen und Fehlermeldungen sein. Um dieses Problem zu lösen, können Sie die Deklaration von **struct** *s* vor die sich auf sie beziehenden Funktions-Prototypen verschieben. Oder Sie fügen die unvollständige Deklaration **struct** *s* vor den sich auf sie beziehenden Funktions-Prototypen ein. Handelt es sich bei dem Funktions-Parameter um eine Struktur und nicht um einen Zeiger auf eine Struktur, dann reicht die unvollständige Deklaration nicht aus. Die vollständige Deklaration der Struktur muß in diesem Fall vor dem Prototypen stehen.

Default argument value redeclared *Compiler-Fehler*

Vorgabewert des Argumentes mehrmals deklariert

Wenn ein Parameter einer C++ Funktion mit einem Vorgabewert deklariert ist, kann dieser Wert bei einer weiteren Deklaration derselben Funktion nicht verändert, nicht nochmals deklariert und auch nicht weggelassen werden.

Default argument value redeclared for parameter *parameter* *Compiler-Fehler*

Vorgabewert des Argumentes für Parameters *Parameter* nochmale deklariert

Wenn ein Parameter einer C++ Funktion mit einem Vorgabewert deklariert ist, kann dieser Wert bei einer weiteren Deklaration derselben Funktion nicht verändert, nicht nochmals deklariert und auch nicht weggelassen werden.

Default expression may not use local variables *Compiler-Fehler*

Standard-Ausdruck kann keine lokalen Variablen benutzen

Ein Ausdruck mit voreingestelltem Argument darf keine lokalen Variablen oder anderen Parameter haben.

Default outside of switch *Compiler-Fehler*

default: nur innerhalb von **switch** zulässig

Eine **default**-Anweisung ist nur innerhalb einer **switch**-Anweisung erlaubt. Hier handelt es sich meistens um eine falsch plazierte geschweifte Klammer.

Default value missing *Compiler-Fehler*

Vorgabe-Wert fehlt

Deklariert eine C++ Funktion einen Parameter mit Vorgabewert, müssen alle folgenden Parameter auch Vorgabewerte haben. In dieser Deklaration war dies nicht der Fall.

Default value missing following parameter *parameter* *Compiler-Fehler*

Vorgabewert: für Parameter *Parameter* fehlt

Die Vorgabewerte aller Parameter, die auf den ersten Parameter mit Vorgabewert folgen, müssen ebenso einzeln angegeben werden.

Define directive needs an identifier *Compiler-Fehler*

Define-Anweisung benötigt einen Bezeichner

Das erste Nicht-Leerzeichen nach **#define** muß ein Bezeichner sein. Der Compiler fand irgendein anderes Zeichen.

symbol **defined in module** *Modul1* **is duplicated in module** *Modul2* *Linker-Fehler*

In Modul *Modul1* definiertes Symbol *Symbol* ist im Modul *Modul2* noch einmal vorhanden

Sie erhalten diese Warnung, wenn es zwischen zwei Symbolen (öffentlichen oder gemeinsamen) zu Konflikten kommt. Dies bedeutet normalerweise, daß ein Symbol in zwei Modulen definiert ist. Ein Fehler tritt dann auf, wenn beide in der (oder den) .OBJ-Datei(en) stehen, da TLINK nicht entscheiden kann, welches Symbol das gültige ist. TLINK gibt eine Warnung aus, wenn eines der duplizierten Symbole in einer Bibliothek und das andere in einer .OBJ-Datei gefunden wird. In diesem Fall verwendet TLINK das Symbol in der .OBJ-Datei.

Delete array size missing] *Compiler-Fehler*

Bei der Angabe der Array-Größe fehlt]

Im Array-Bezeichner eines Operators fehlt eine eckige rechte Klammer

Destructor cannot be declared const or volatile *Compiler-Fehler*

Deklaration des Destruktors als **const** oder **volatile** nicht erlaubt

Ein Destruktor wurde als **const** und/oder **volatile** deklariert. Dies ist nicht erlaubt.

Destructor cannot have a return type specification *Compiler-Fehler*

Destruktor darf keinen Rückgabetyp besitzen

C++ Destruktoren können niemals einen Wert zurückgeben, und so darf auch der Typ des Rückgabewerts nicht festgelegt werden.

Destructor for *Klasse* **is not accessible** *Compiler-Fehler*

Destruktor für class nicht zugänglich

Der Destruktor für diese C++ Klasse ist **protected** oder **private** und kann hier nicht verwendet werden, um die Klasse zu löschen. Ist ein Klassendestruktor **private**, kann die Klasse nicht zerstört und daher auch nie benutzt werden - dies ist vermutlich ein Programmierfehler. Auf einen **protected**-Destruktor kann nur über abgeleitete Klassen zugegriffen werden. Dies ist eine sinnvolle Methode, um sicherzustellen, daß keine Instanzen einer Basisklasse, sondern nur abgeleitete Klassen erzeugt werden.

Destructor for class required in conditional expression *Compiler-Fehler*

In bedingtem Ausdruck ist ein Destruktor für *Klasse* nötig

Wenn in einem bedingten Ausdruck der Compiler eine temporäre lokale Variable erzeugen muß, kann der Destruktor möglicherweise nicht aufgerufen werden, weil die Variable nicht initialisiert ist. Die temporäre Variable kann explizit mit Klassennamen(val, val) oder implizit durch andere Codes erzeugt werden. Überarbeiten Sie Ihren Quellcode, um die temporäre Variable zu entfernen.

Destructor name must match the class name *Compiler-Fehler*

Name des Destruktors muß zur Klasse passen

In einer C++ Klasse führt die Tilde (~) die Deklaration des Klassendestruktors an. Dieser muß genauso heißen wie die Klasse. In Ihrer Quelldatei steht die Tilde vor einem anderen Namen.

Divide error *Laufzeitfehler*

Division durch 0

Sie haben versucht, eine Integerzahl durch Null zu dividieren, was nicht erlaubt ist, z.B.:

```
int n = 0;
n = 2 / n;
```

Sie können diesen Fehler mit einer Signal-Funktion bahandeln. Andernfals ruft Borland C++ *abort* auf und Ihr Programm terminiert.

Division by zero *Compiler-Fehler*

Division durch Null

Ein Ausdruck in Ihrer Quelldatei enthält eine Division durch Null.

Division by zero *Compiler-Warnung*

Division durch Null

Ein Divisions- oder Restoperator mit einem konstanten Ausdruck, der eine Null im Nenner enthielt.

Division by zero *Compiler-Warnung*

Division durch Null

Ein Divisions- oder Restausdruck hatte eine Null im Nenner.

Division by zero *MAKE-Fehler*

Division durch Null

Ein Divisions- oder Restausdruck in einer **!if**-Anweisung hatte eine Null im Nenner.

do statement must have while *Compiler-Fehler*

do-Anweisung muß **while** enthalten

Einer **do**-Schleife fehlt das abschließende **while**.

Dateiname **does not exist - don't know how to make it** *Make-Fehler*

Dateiname nicht vorhanden - Ich weiß nicht, wie ich sie anlegen soll

Es trat ein nichtexistierender Dateiname während der Generierung auf und es gibt keine Regel, diese Datei zu erzeugen.

DOS error, ax = N*ummer* *Linker-Fehler*

DOS-Fehler, ax = *Nummer*

Dieser Fehler tritt auf, wenn ein DOS-Aufruf einen unerwarteten Fehler zurücklieferte. Der ausgegebene Wert von ax ist der resultierende Fehlercode. Dies könnte auf einen internen Fehler von TLINK oder auf einen DOS-Fehler hinweisen. Die einzigen Aufrufe von TLINK an DOS, bei denen dieser Fehler auftreten könnte, sind read, write, seek und close.

do while statement missing (*Compiler-Fehler*
In einer **do while**-Anweisung fehlt **(**
In einer **do**-Anweisung fand der Compiler nach dem Schlüsselwort **while** keine öffnende runde Klammer.

do-while statement missing) *Compiler-Fehler*
In einer **do while**-Anweisung fehlt **)**
In einer **do while** Anweisung fand der Compiler nach der Bedingung keine schließende runde Klammer.

do-while statement missing ; *Compiler-Fehler*
In einer **do while**-Anweisung fehlt **;**
In einer **do while**-Bedingung fand der Compiler nach der abschließenden runden Klammer kein Semikolon.

Duplicate case *Compiler-Fehler*
case-Marke doppelt vorhanden
Der Wert einer jeden **case**-Marke einer **switch**-Anweisung darf nur einmal vorkommen.

Duplicate Handler for *Typ1*, **already had** *Typ2* *Compiler-Fehler*
Doppelte Behandlungsroutine für *Typ1*, bereits aufgetreten für *Typ2*
Es ist nicht zulässig, zwei Behandlungsroutinen für den selben Typ anzugeben.

Duplicate ordinal for exports: string *(ordval1)* **and string** *(ordval2)* *Linker-Fehler*
Doppelter Ordinalwert für Exporte: Zeichenkette (*Ordinalwert1*) und Zeichenkette (*Ordinalwert2*)
Zwei Export-Anweisungen haben das gleiche Symbol, aber verschiedene Ordinalwerte. Sie müssen den gleichen Ordinalwert benutzen oder eine der Export-Anweisungen löschen.

Empty LEDATA record in module *Modul* *Linker-Warnung*
Leerer LEDATA-Record in Modul *Modul*
Diese Warnung tritt auf, wenn der Übersetzer einen Datenrecord aussondert, der Daten enthält. Wenn dieses passieren sollte, teilen Sie dieses Ihrem Fachhändler mit. Es sollten keine negativen Seiteneffekte durch den Record entstehen.

Enum syntax error *Compiler-Fehler*
Syntax-Fehler bei Aufzählung
Eine Deklaration mit **enum** hat eine ungültige Syntax. Die Bezeichnerliste in einer **enum**-Deklaration ist nicht korrekt.

Error changing file buffer size *Bibliotheken-Fehler*
Fehler beim Verändern der Puffer-Größe
Eine Bibliotheksverwaltung versucht durch Lesen oder Schreiben in einem File die Größe eines Puffers zu verändern. Es ist aber nicht genug Speicherplatz vorhanden. Möglicherweise kann dieses Problem bereits durch ein wenig mehr System-Speicher gelöst werden.

Error directive: *Meldung* *Compiler-Fehler*
Fehler-Anweisung: *Meldung*
Der Text der **#error**-Anweisung, die in der Quelldatei ausgeführt wurde, wird angezeigt.

Error directive: *Meldung* *MAKE-Fehler*
Fehler-Anweisung: *Meldung*
MAKE führte eine Anweisung in der Quelldatei aus. Der Text dieser Anweisung wird in der Meldung angezeigt.

Error opening *Dateiname* *Bibliotheken-Fehler*
Fehler beim Öffnen von *Dateiname*
TLIB kann die angegebene Datei aus irgendeinem Grund nicht öffnen.

Error opening *Dateiname* **for output** *Bibliotheken-Fehler*

Fehler beim Öffnen der Datei *Dateiname*

Die Bibliotheksverwaltung konnte die angegebene Ausgabedatei nicht öffnen. Das kann an mangelndem Speicherplatz für die Ziel-Bibliothek oder einer Auflistungs-Datei liegen. Dieser Fehler tritt auf, wenn die Zieldatei zwar existiert, aber als Nur-Lese-Datei gekennzeichnet ist.

Error renaming *Dateiname1* **to** *Dateiname2* *Bibliotheken-Fehler*

Fehler beim Umbenennen von *Dateiname1* in *Dateiname2*

Dieser Fehler tritt auf, wenn TLIB eine temporäre Bibliotheksdatei erzeugt und die temporäre Datei den Namen der Zielbibliothek erhält. Dieser Fehler deutet auf mangelnden Plattenspeicher hin.

Error writing output file *Compiler-Fehler*

Fehler beim Schreiben der Ausgabedatei

Es liegt ein DOS-Fehler vor, der Borland C++ daran hindert, eine OBJ-, EXE- oder temporäre Datei zu erzeugen. Lösungen: Überprüfen Sie, ob das Ausgabeverzeichnis tatsächlich vorhanden ist. Überprüfen Sie, ob Ihre Diskette bzw. Platte über genügend freien Speicherplatz verfügt.

__except **or** *__finally* **expected following** *__try* *Compiler-Fehler*

_except oder *_finally* mit nachfolgendem *__try* erwartet

In C muß auf einen *_try*-Block ein *_except*- oder ein *_finally*-Behandlungs-Block folgen.

Exception handling variable may not be used here *Compiler-Fehler*

Variable für Ausnahmebehandlung darf hier nicht verwendet werden

Es wurde versucht einen Wert der Ausnahmebehandlung zu benutzen, der ausschließlich besonderen Konstrukten zur Ausnahmebehandlung vorbehalten ist, z.B. GetExceptionCode().

Exception specification not allowed here *Compiler-Fehler*

Ausnahmespezifizierung hier nicht erlaubt

Deklarationen von Zeigern auf Funktionen dürfen keine Ausnahmespezifizierungen enthalten.

Explicit stacks are ignored for PE images *Linker-Warnung*

Explizite Stacks werden für PE-Programme ignoriert

Win32-Anwendungen sind Anwendungen im PE-Format, die keine expliziten Stacks besitzen. Das Stacksegment wird in das Programm eingebunden, aber es wird nicht als Stack für die Anwendung verwendet. Stattdessen wird der Parameter Stackgröße benutzt, um den Stack auf die entsprechende Größe zu setzen, und das Betriebssystem legt den Stack für die Anwendung an.

Export *symbol* **is already imported** *Linker-Warnung*

Das exportierte *symbol* wird bereits importiert.

Sie haben versucht, ein Symbol zu exportieren, das sie bereits importieren.

Export *symbol* **is duplicated** *Linker-Warnung*

Export *symbol* ist doppelt

Diese Warnung erscheint, wenn zwei unterschiedliche Funktionen mit demselben Namen über **_export** exportiert werden. TLINK entscheidet sich in diesem Fall für die erste Definition.

Expression expected *Compiler-Fehler*

Ausdruck erwartet

Hier wird ein Ausdruck erwartet, der mit dem aktuellen Zeichen aber nicht beginnen kann. Dieser Fehler tritt eventuell beim Steuerausdruck für eine **if**- oder **while**-Anweisung oder beim Initialisieren einer Variablen auf, oft wegen eines versehentlich eingefügten oder gelöschten Zeichens.

Expression of scalar type expected *Compiler-Fehler*

Skalarer Ausdruck erwartet

Die Operatoren not (**!**), Increment (**++**) und Dekrement (**--**) verlangen einen skalaren Ausdruck. Erlaubt sind neben Zeigern nur die Typen **char**, **short**, **int**, **long**, **enum**, **float**, **double** und **long double**.

Expression syntax *Compiler-Fehler*

Syntax-Fehler

Dies ist eine allgemeine Fehlermeldung, wenn der Compiler einen Ausdruck nicht auswerten kann. Mögliche Ursachen:

- Zwei aufeinanderfolgende Operatoren
- Falsche oder fehlende Klammer
- Fehlendes Semikolon in der vorhergehenden Anweisung

Expression syntax error in !if statement *MAKE-Fehler*

Syntax-Fehler im Ausdruck einer **!if**-Anweisung

Der Ausdruck in einer **!if**-Anweisung ist fehlerhaft: Er enthält eine Klammer an der falschen Stelle, oder einen Operator bzw. eine Konstante zuviel oder zuwenig.

Grund **- extended dictionary not created** *Bibliotheken-Fehler*

Grund - erweitertes Verzeichnis nicht erzeugt

TLIB konnte aufgrund der in der Warnung angegebenen Ursachen das erweiterte Verzeichnis nicht erstellen.

Extended dictionary not found in library *Bibliothekname* **/E option ignored** *Linker-Warnung*

Das erweiterte Verzeichnis konnte in Bibliothek *Bibliothekname* nicht gefunden werden: Erweitertes Verzeichnis wird ignoriert,

Die Option **/E** benötigt, daß alle am Link durch TLINK beteiligten Bibliotheken erweiterte Verzeichnisse haben. Beim Linkvorgang war eine nicht erweiterte Bibliothek beteiligt, während die Option **/E** gesetzt war. Der Linker unterbrach den Vorgang und fuhr mit einem Standard-Link fort.

Extern variable cannot be initialized *Compiler-Fehler*

Externe Variable kann nicht initialisiert werden

Die Speicherklasse **extern** bedeutet für eine Variable, daß sie hier deklariert, aber nicht definiert wird und kein Speicherplatz für sie vorgesehen ist.Deshalb kann sie in der Deklaration nicht initialisiert werden.

Extern symbol was not qualified with import in module *Modul* *Linker-Warnung*

Extern *symbol* ist nicht mit _import im Modul *Modul* gekennzeichnet

Win32-Anwendungen, die sich auf importierte Symbole beziehen, benötigen Indirektionen, um auf Daten zuzugreifen. Bei Aufrufen, wird dieses automatisch von TLINK gehandhabt. Für Referenzen auf importierte Daten, muß der Compiler eine Indirektion erzeugen, oder die Anwendung läuft nicht korrekt. Der Compiler kann nur dann eine Indirektion erstellen, wenn das Symbol mit _import gekennzeichnet ist. Wenn TLINK eine segmentrelative Referenz zu einem importierten Symbol findet, und dieses nicht mit _import versehen ist, so erscheint die Warnung.

Extra argument in template class name *template* *Compiler-Fehler*

Überzähliges Argument im Namen der Template-Klasse *template*

Der Klassenname eines Template gab zu viele aktuelle Werte für ihre formalen Parameter an.

Extra parameter in call *Compiler-Fehler*

Überzähliger Parameter bei Funktionsaufruf

Der Aufruf einer Funktion, die über einen Zeiger auf einen Prototypen definiert wurde, erfolgte mit zu vielen Parametern.

Extra parameter in call to *function* *Compiler-Fehler*

Überzähliger Parameter bei Aufruf der Funktion *Function*

Der Aufruf einer Funktion, die bereits über einen Prototypen definiert wurde, erfolgte mit zu vielen Parametern.

Failed read from *Dateiname* *Linker-Fehler*

Lesevorgang von Datei *Dateiname* fehlgeschlagen

TLINK konnte aus der Datei nicht lesen.

Failed to locate DPMI server (DPMI16BI.OVL) *Compiler-Fehler*

Konnte den DPMI-Server (DPMI16BI.OVL) nicht finden

Failed to locate protected mode loader (DPMILOAD.EXE) *Compiler-Fehler*

Konnte die Ladedatei für den Protected Mode (DPMILOAD.EXE) nicht finden

Stellen Sie sicher, daß sich die Dateien DPMI16BI.OVL und DPMILOAD.EXE an der richtigen Stelle befinden: Entweder in Ihrem Pfad oder im Verzeichnis des Kommando-Zeilen-Dienstprogramms, das Sie benutzen wollen.

Failed write to *Dateiname* *Linker-Fehler*

Schreibvorgang auf die Datei *Dateiname* fehlgeschlagen

TLINK konnte nicht in die Datei schreiben.

__far16 may only be used with __pascal or __cdecl *Compiler-Fehler*

__far16 kann nur mittels **__pascal** oder **__cdecl** genutzt werden

Sie benutzen **__far16**, um in einer 16-bit DLL Funktionen aufzurufen oder Daten zu referenzieren. Diese Funktionen bzw. Daten können nur mittels **__pascal** oder **__cdecl** verändert werden.

File must contain at least one external declaration *Compiler-Fehler*

Datei muß mindestens eine externe Deklaration enthalten

Der Inhalt dieser Compilierungseinheit war logisch leer, weil sie keine Extern-Deklaration enthielt. ANSI C und C++ legen fest, daß etwas in der zu compilierenden Einheit deklariert werden muß.

Filename too long *Compiler-Fehler*

Dateiname in **#include**-Anweisung zu lang

Der Compiler kann den Dateinamen in einer **#include**-Anweisung nicht bearbeiten. Pfadnamen dürfen nur bis zu 260 Zeichen lang sein.

File name too long *MAKE-Fehler*

Dateiname in einer **!include**-Anweisung zu lang

Der Dateiname der **!include**-Anweisung war zu lang. Der 512 Byte große interne Puffer von MAKE ist deshalb übergelaufen.

Dateiname **file not found** *Bibliotheken-Warnung*

Datei *Dateiname* nicht gefunden

TLIB versucht bei einem Befehl der Kommandozeile ein nicht existierendes Objekt hinzuzufügen. Die Bibliothek wurde trotzdem angelegt.

Dateiname **file not found** *Bibliotheken-Fehler*

Datei *Dateiname* nicht gefunden

Die IDE erzeugt die Bibliothek, indem die bestehende Bibliothek entfernt und neu erstellt wird. Falls eines der angegebenen Objekte nicht existiert, ist die Bibliothek unvollständig und dieser Fehler wird generiert.

Wenn die IDE das Fehlen eines Objekts anzeigt, wurde entweder das Quelltext-Modul nicht compiliert oder es sind Fehler bei der Compilierung aufgetreten. Die Verwendung der Befehle *Projekt/Compilieren* oder *Projekt/Make-Datei erzeugen* sollte das Problem beheben oder auf den Ursprung des Fehlers hindeuten.

Dateiname **(***Zeilennummer***): file read error** *Linker-Fehler*

Dateiname (*Zeilennummer*): Fehler beim Lesen der Datei

Beim Lesen der Moduldefinitionsdatei durch TLINK trat ein DOS-Fehler auf. Normalerweise bedeutet dies ein vorzeitiges Erreichen des Dateiendes.

Fixup to zero length segment in module *Modul* *Linker-Fehler*

Relatives Fixup erzeugt einen Überlauf in Modul *Modul*

Es wurde nach dem Ende eines Image-Segmentes eine Referenz gemacht. Diese Referenz würde mit der Erzeugung einer ungültigen Adresse enden und wurde als Fehler angezeigt.

Fixup overflow at segment:xxxh, target = segment:xxxh in module *modul* *Linker-Warnung*

Fixup-Überlauf bei Segment:xxxxh, Ziel = *Segment*:xxxh im Modul *Modul*

Fixup overflow at segment:xxxh, target = *Symbol* **in module** *Modul* *Linker-Warnung*

Fixup-Überlauf bei Segment:xxxxh, Ziel = *Symbol* im Modul *Modul*

Diese Meldungen signalisieren eine falsche Code- oder Datenreferenzierung in einer Objektdatei, die TLINK zur Linkzeit adressiert.

Fixup-Überlauf bei Segment:xxxxh, Ziel = *Segment*:xxxh im Modul *Modul*

Diese Meldung tritt sehr häufig bei unpassenden Speichermodellen auf. Der häufigste Grund ist ein **near**-Aufruf einer Funktion, die in einem anderen Code-Segment steht.

Fixup-Überlauf bei Segment:xxxxh, Ziel = *Symbol* im Modul *Modul*

Dieser Fehler kann auch auftreten, wenn Sie einen **nea**r-Aufruf für eine Datenvariable oder eine Datenreferenz für eine Funktion generieren.

In beiden Fällen ist das in der Fehlermeldung als Ziel benannte Symbol die referenzierte Variable oder Funktion. Die Referenz liegt in dem angegebenen Modul. Sehen Sie sich die Quelltextdatei des Moduls an, um die Referenz zu korrigieren. Führen Sie nochmals ein *Erzeugen* für das gesamte Programm durch, um sicherzustellen, daß alle Module unter demselben Speichermodell compiliert wurden. Überprüfen Sie, daß eingebundene Bibliotheken ebenfalls unter demselben Speichermodell compiliert wurden. Wenn diese Vorgehensweise nicht zum Aufdecken der Fehlerursache führt oder wenn Sie mit Assembler oder einer anderen Hochsprache neben C oder C++ programmieren, kann es auch andere Ursachen für diese Meldung geben. In C++ könnte diese Meldung auch dann auftreten, wenn Sie nicht die für ein gegebenes Speichermodell standardmäßig eingestellten Werte, sondern andere Segment- oder Gruppennamen verwenden.

Fixup to zero length segment in module *Modulname* *Linker-Fehler*

Fixup auf ein Segment mit Länge null in Modul *Modulname*

Es wurde ein Verweis hinter das Ende eines Programmsegments gemacht. Dieses würde zum Zugriff auf eine ungültige Adresse führen. Daher wurde die Referenz als fehlerhaft markiert.

Floating point error: Divide by 0. *Laufzeitfehler*

Gleitpunkt-Fehler: Teilen durch 0

Floating point error: Domain. *Laufzeitfehler*

Gleitpunkt-Fehler: Wertebereich

Floating poing error: Overflow. *Laufzeitfehler*

Gleitpunkt-Fehler: Überlauf

Diese fatalen Fehler ergaben sich aus einer Gleitpunkt-Operation deren Ergebnis unendlich ist, wie z.B. bei 1.0/0.0.
»Teilen durch 0« bedeutet, daß das Ergebnis plus oder minus unendlich ist.
»Wertebereich« bedeutet, daß das Ergebnis keine Zahl ist.
»Überlauf« bedeutet, daß das Ergebnis bei einem völligen Verlust der Genauigkeit plus oder minus Unendlich war. Ein Beispiel für diesen Fall ist die Zuweisung von 1e200*1e200 an den Typ **double**.

Floating point error: Partial loss of precision. *Laufzeitfehler*

Gleitpunkt-Fehler: Partieller Verlust der Genauigkeit

Floating point error: Underflow. *Laufzeitfehler*

Gleitpunkt-Fehler: Unterlauf underflow-Unterlauf ?

Diese Ausnahmen sind standardmäßig maskiert und die Fehlermeldungen werden nicht erkannt. Unterläufe werden mit 0 bewertet und der Genauigkeitsverlust wird ignoriert. Sie können durch den Aufruf von *_control87* demaskiert werden.

Floating point error: Stack fault. *Laufzeitfehler*

Gleitpunkt-Fehler: Stack-Fehler

Der stack für Gleitpunktzahlen ist übergelaufen. Dieser Fehler wird normalerweise entdeckt. Er tritt auf, wenn Assembler-Code zu viele Register benutzt, oder wenn eine Gleitkomma-Funktion falsch deklariert wurde.

Diese Gleitpunkt-Fehler können vermieden werden: Die Ausnahmebehandlung kann maskiert werden, so daß sie nicht auftritt oder sie kann mittels *signal* abgefangen werden. Für genauere Informationen siehe die Funktionen *_control87* und *signal*. In jedem der obigen Fälle zeigt das Programm eine Fehlermeldung an. Dann wird *abort* aufgerufen, zeigt

```
Abnormal program termination
```

an und ruft wiederum exit(3) auf. Für genauere Informationen siehe abort und exit.

for statement missing (*Compiler-Fehler*
In einer **for**-Anweisung fehlt **(**
In einer **for**-Anweisung fand der Compiler nach dem Schlüsselwort **for** keine schließende runde linke Klammer.

for statement missing (*Compiler-Fehler*
In einer **for**-Anweisung **)**
In einem **for**-Ausdruck fand der Compiler nach dem Kontroll-Ausdruck keine schließende rechte runde Klammer.

for statement missing ; *Compiler-Fehler*
In einer **for**-Anweisung fehlt **;**
In einer **for**-Anweisung fand der Compiler nach einem der Ausdrücke kein Semikolon.

Friends must be functions or classes *Compiler-Fehler*
Friends müssen Funktionen oder Klassen sein
Ein **friend** einer C++ Klasse muß entweder eine Funktionen oder eine Klasse sein.

Function call missing) *Compiler-Fehler*
Funktionsaufruf: nach Parameterliste fehlt **)**
Die Parameterliste des Funktionsaufrufes weist einen Syntaxfehler auf, z.B. eine nicht abgeschlossenen Klammerebene.

Function calls not supported *Compiler-Fehler*
Funktionsaufrufe werden nicht unterstützt
In der integrierten Ausdrucksauswertung des Debuggers werden Funktionsaufrufe nicht unterstützt. Darunter fallen implizite Konvertierungen von Funktionen, Konstruktoren, Destruktoren, überladene Operatoren und **inline** -Funktionen.

Function defined inline after use as extern *Compiler-Fehler*
Funktion soll nach externer Verwendung als **inline** deklariert werden.
Nach der Deklaration einer Funktion als extern kann diese nicht mehr als **inline** deklariert werden. Verlagern Sie die **inline**-Deklaration im Quelltext weiter nach vorne oder löschen Sie sie ganz.

Function definition cannot be a Typedef'ed declaration *Compiler-Fehler*
Funktions-Deklaration darf keine **typedef**-Deklaration sein.
Im ANSI C Standard darf ein Funktionsrumpf nicht durch ein **typedef** mit einem Funktions-Typen definiert werden.

Function *function* **cannot be static** *Compiler-Fehler*
Die Funktion *Funktion* darf nicht **static** sein
Nur gewöhnliche Elementfunktionen und die Operatoren **new** und **delete** können **static** deklariert werden. Konstruktoren, Destruktoren und andere Operatoren dürfen nicht **static** sein.

Function *function* **should have a prototype** *Compiler-Fehler*
Funktion *Funktion* sollte einen Prototyp haben
Eine Funktion ohne Prototyp im Gültigkeitsbereich wurde aufgerufen.
In C ist **int** *foo*(); kein Prototyp, während **int** *foo* **int**) und **int** *foo* (**void**); Prototypen darstellen. In C werden Prototypen für alle Funktionen empfohlen. In C++ sind Prototypen für alle Funktionen erforderlich. In allen Fällen dient eine Funktionsdefinition (ein Funktionskopf mit Rumpf) als Prototyp, wenn sie die erste Erwähnung dieser Funktion ist.

Function should return a value *Compiler-Warnung*
Funktion sollte einen Wert zurückliefern

Die Funktion ist so deklariert, daß sie einen anderen Wert als **int** oder **void** zurückliefert. Der Compiler hat aber keinen Rückgabewert vorgefunden. **int**-Funktionen sind ausgenommen, da es in alten C-Versionen keinen **void**-Typ gab. Diese Meldung können Sie mit der Kommandozeilenoption **-wrvl** kontrollieren. Nach der Standardeinstellung wird die Meldung angezeigt.

Functions *function1* **and** *function2* **both use the same dispatch number** *Compiler-Fehler*

Die Funktionen *function1* und *function2* verwenden beide die selbe Zuordnungsnummer.
Dieser Fehler deutet auf ein Problem mit einer dynamisch zugeordneten, virtuellen Tabelle hin (DDVT). Wenn Sie eine dynamisch zugeordnete Funktion in einer abgeleiteten Klasse überschreiben, benutzen Sie den gleichen zugeordneten Index. Innerhalb einer Klassenhierarchie muß jede Funktion einen anderen zugeordneten Index enthalten.

Functions cannot return arrays or functions *Compiler-Fehler*

Funktion kann keine Felder oder Funktionen zurückgeben
Sie haben eine Funktion definiert, die ein Feld oder eine Funktion zurückgibt. Überprüfen Sie, ob entweder ein Zeiger auf ein Feld oder eine Funktion (vielleicht fehlt *) zurückgegeben werden sollte, oder ob die Funktionsdefinition einen falschen Datentyp enthält.

Functions containing local destructors are not expanded inline in function *Funktion* *Compiler-Warnung*

Funktion *Funktion* enthält lokale Destruktoren und wird nicht **inline** ausgeführt
Sie haben eine **inline**-Funktion programmiert, für die Borland C++ das **inline** ausgeschaltet hat. Sie können diese Warning ignorieren, wenn sie wollen: Die Funktion wird dann als outline erzeugt.

Functions containing *reserved word* **are not expanded inline** *Compiler-Warnung*

Funktionen mit reservierten Wörtern können nicht **inline** erweitert werden oder Funktionen mit lokalen Destruktoren werden in der Funktion *Funktion* nicht **inline** erweitert
Reservierte Wörter:
Funktionen, welche eines oder mehrere dieser reservierten Wörter enthalten, können auch dann nicht **inline** erweitert werden, wenn sie **inline** spezifiziert sind:

- **break**
- **case**
- **continue**
- **do**
- **for**
- **goto**
- **switch**
- **while**

Die Funktion ist immer noch zulässig, wird jedoch **static** (nicht global) behandelt. Eine Kopie dieser Funktion erscheint in jeder Compilierungseinheit, in der sie aufgerufen wird.
Lokale Destruktoren: Sie haben eine **inline**-Funktion erstellt, für welche durch den Compiler **inline** außer Kraft gesetzt ist. Sie können diese Warnung übergehen; die Funktion wird ohne **inline** generiert. Die Kommandozeilenoption **-winl** bestimmt über die Ausgabe der Warnung. Standardmäßig ist sie eingeschaltet.

Functions may not be part of a struct or union *Compiler-Fehler*

Funktionen dürfen nicht Teil von **struct** oder **union** sein
Hier wurde ein Feld eines **struct** oder **union** als Funktion anstatt als Zeiger auf eine Funktion deklariert. Nur in C++ sind Felder als Funktionen erlaubt.

General error *Linker-Fehler*
General error in library file *Dateiname* **in module modul near** *module* **offset 0xyyyyyyyy** *Linker-Fehler*
General error in module *modul* **near module file offset 0xyyyyyyyy** *Linker-Fehler*

Allgemeiner Fehler ODER allgemeiner Fehler in Bibliotheks-Datei

Dateiname im Modul *Modul* bei Moduldatei-Offset *0xyyyyyyyy* ODER allgemeiner Fehler im Modul *Modul* bei Moduldatei-Offset *0xyyyyyyyy*. TLINK gibt so viele Informationen als möglich über den Bearbeitungsschritt an, der beim Auftreten des unbekannten Fehlers mit sofortigem Abbruch gerade stattfand.

General error
General error in module *Modul* — *Linker-Fehler*

Allgemeiner Fehler in Modul *Modul*
Eine nicht abgefangene Ausnahme ist in TLINK aufgetreten.

Verständigen Sie Borland Tech Support.

Global anonymous union not static — *Compiler-Fehler*

Globale anonyme **Union** nicht **static**
In C++ muß eine globale, anonyme **Union** auf Dateiebene **static** sein.

Goto bypasses initialization of a local variable — *Compiler-Fehler*

Goto umgeht die Initialisierung einer lokalen Variable
Es ist in C++ nicht zulässig, die Initialisierung einer lokalen Variable zu umgehen. Dieser Fehler deutet auf eine **goto**-Anweisung hin, welche die Steuerung hinter diese lokale Variable transferiert.

Goto Into an Exception Handler is Not Allowed — *Compiler-Fehler*

Einsprung in eine Ausnahmebehandlungsroutine ist nicht zulässig
Ein Sprung in einen **try**-Block oder eine Ausnahmebehandlungsroutine, die einem **try**-Block zugeordnet ist, ist nicht zulässig.

Goto statement missing label — *Compiler-Fehler*

goto-Befehl: kein Sprungziel angegeben
Nach dem reservierten Wort **goto** muß die Angabe eines Bezeichners folgen.

Group *Gruppe* **exceeds 64K** — *Linker-Fehler*

Gruppe *Gruppe* größer als 64 KByte
Diese Meldung erscheint, wenn bei der Kombination der Gruppensegmente 64 KByte überschritten werden.

Group overflowed maximum size: *Gruppe* — *Compiler-Fehler*

Gruppe *Gruppe* überschritt die maximale Größe
Die Gesamtgröße der Segmente einer Gruppe (z.B.DGROUP) überstieg 64KByte

Group *Gruppe1* **overlaps group** *Gruppe2* — *Linker-Warnung*

Gruppe *Gruppe1* überschneidet Gruppe *Gruppe2*
TLINK hat verschachtelte Gruppen gefunden.

Handler for *Typ1* **hidden by previous handler for** *Typ2* — *Compiler-Warnung*

Behandlungsroutine für *Typ1* durch vorhergehende Routine für *Typ2* verdeckt
Diese Warnung erscheint, wenn eine Behandlungsroutine für einen Typ *D*, der von einem Typ *B* abgeleitet ist, nach der Routine für *B* spezifiziert wird. Die Routine für *D* wird niemals aktiviert.

Spezifizierer **has already been included** — *Compiler-Fehler*

Spezifizierer schon vorhanden
Der Spezifizierer taucht in der Deklaration mehr als einmal auf.
Löschen oder ändern Sie einen Spezifizierer.

Hexadecimal value contains more than 3 digits — *Compiler-Warnung*

Hexadezimaler Wert hat mehr als drei Stellen
In älteren C-Versionen konnten hexadezimale Escape-Sequenzen nicht mehr als drei Stellen lang sein. Der ANSI-Standard erlaubt beliebig viele Stellen, solange der Wert noch in ein Byte paßt. Diese Warnung erscheint bei einer langen hexadezimalen Escape-Sequenz mit vielen führenden Nullen (z.B. »\x00045«). Ältere C-Versionen interpretieren diesen String unterschiedlich.

Funktion1 **hides virtual function** *Funktion2* — *Compiler-Warnung*

Funktion1 verdeckt virtuelle Funktion *Funktion2*

Eine virtuelle Funktion wird üblicherweise von der Deklaration einer abgeleiteten Klasse überschrieben. In diesem Fall ist weiteren abgeleiteten Funktionen durch eine Deklaration mit demselben Namen, aber mit unterschiedlichen Argumenttypen, der Zugriff auf die virtuellen Funktionen nicht mehr möglich. Mit der Kommandozeilenoption **-whid** können Sie die Meldung kontrollieren. Standardmäßig ist sie eingeschaltet.

Identifier expected — *Compiler-Fehler*

Bezeichner erwartet

Hier wurde ein Bezeichner erwartet, aber nicht gefunden.

In C wird in folgenden Situationen ein Bezeichner erwartet:

- bei einer Parameterliste in einem Funktionskopf im alten Stil,
- nach den reservierten Worten **struct** und **union**, wenn keine Klammern folgen, und
- als Element in einer Struktur oder **union** (außer bei Bitfeldern der Breite 0).

In C++ wird zusätzlich in diesen Situationen ein Bezeichner erwartet:

- wenn eine Liste von Basisklassen auftritt, von der eine weitere Klasse abgeleitet wird, und zwar nach dem doppelten Doppelpunkt (**::**), und
- nach dem reservierten Begriff **operator**, wenn kein Operator folgt.

Identifier *Bezeichner* **cannot have a type qualifier** — *Compiler-Fehler*

Bezeichner *Bezeichner* darf keinen Typ-Qualifizierer haben

Der C++ Qualifizierer Klass **::**Bezeichner ist hier nicht erlaubt.

Ein Qualifizierer ist nicht erlaubt bei

- **typedef**-Namen,
- Funktionsdeklarationen (außer Definitionen auf Dateiebene),
- vor lokalen Variablen oder Funktionsparametern und
- bei Klassenelementen, außer als Qualifizierer der eigenen Klasse, was überflüssig, aber zulässig ist.

If statement missing (— *Compiler-Fehler*

In einer **if**-Anweisung fehlt **(**

In einer **if**-Anweisung fand der Compiler nach dem Schlüsselwort **if** keine öffnende runde Klammer.

If statement missing) — *Compiler-Fehler*

In einer **if**-Anweisung fehlt **)**

In einer **if**-Anweisung fand der Compiler nach der Bedingung keine abschließende runde Klammer.

If statement too long — *MAKE-Fehler*

if-Anweisung zu lang

Ifdef statement too long — *MAKE-Fehler*

Ifdef-Anweisung zu lang

Ifndef statement too long — *MAKE-Fehler*

Ifndef-Anweisung zu lang

Eine **If**, **Ifdef** oder **Ifndef**-Anweisung überschritt die maximal mögliche Länge von 4096 Zeichen.

Ignored *Modul*, **path is too long** — *Bibliotheken-Warnung*

Modul *Modul* ignoriert, Pfad zu lang

Dieser Fehler erscheint, wenn der *Pfad* einer Bibliotheks- oder Modul-Datei länger als 64 Zeichen ist.

Illegal ACBP byte in SEGDEF in module *Modul* — *Linker-Fehler*

Ungültiges ACBP Byte in SEGDEF im Modul *Modul*

Dieser Fehler ist ein typischer Übersetzerfehler.

Illegal character *character* **(0x***value***)** — *Compiler-Fehler*

Nicht erlaubtes Zeichen *Zeichen* (0x*Wert*)

Der Compiler entdeckte einige ungültige Zeichen in der Eingabedatei. Der Hexadezimalwert des betreffenden Zeichens wird ausgegeben. Dieser Fehler kann auch durch die Übergabe eines zusätzlichen Parameters an ein Funktionsmakro ausgelöst werden.

Illegal character in constant expression *Ausdruck* *MAKE-Fehler*
Nicht erlaubtes Zeichen im konstanten Ausdruck *Ausdruck*
Make hat ein Zeichen gefunden, das in konstanten Ausdrücken nicht erlaubt ist. Wenn es sich bei diesem Zeichen um einen Buchstaben handelt, haben Sie sich vermutlich bei der Angabe eines Bezeichners vertippt.

Illegal component to GRPDEF in module *Modul* *Linker-Fehler*
Ungültige Komponente von GRPDEF im Modul *Modul*
Dieser Fehler ist ein typischer Übersetzerfehler.

Illegal group definition: *Gruppe* **in module** *Modul* *Linker-Fehler*
Fehlerhafte Gruppendefinition: Gruppe *Gruppe* im Modul *Modul*
Dieser Fehler wird von einem unkorrekten GRPDEF-Record in einer OBJ-Datei ausgelöst. Dies kann an selbsterstellten OBJ-Dateien oder an einem Fehler im für die Erstellung der.OBJ-Datei verwendeten Compiler liegen. Tritt dies bei einer mit einem Borland-Compiler erzeugten Datei auf, führen Sie die Compilierung noch einmal durch. Tritt der Fehler weiterhin auf, dann setzen Sie sich mit Borland in Verbindung.

Illegal initialization *Compiler-Fehler*
Nicht zulässige Initialisierung
In C gibt es genau zwei Möglichkeiten der Initialisierung: Die erste ist mittels eines konstanten Ausdrucks. Die zweite ist die Adresse einer globalen Variablen mittels **extern** oder **static** plus oder minus einer Konstante.

Illegal octal digit *Compiler-Fehler*
Unzulässige Zahl in oktaler Konstante
In oktalen Konstanten sind nur 0..7 als Ziffern erlaubt.

Illegal parameter to __emit__ *Compiler-Fehler*
Unzulässiger Parameter für __emit__
Beim direkten Einfügen von Literalen in Ihr Programm mit der Funktion __emit__ gibt es einige Einschränkungen. Sie können beispielsweise eine lokale Variable nicht als Parameter an __emit__ übergeben.

Illegal pointer subtraction *Compiler-Fehler*
Nicht zulässige Subtraktion von Zeigern
Sie haben versucht, einen Zeiger von einem Nicht-Zeiger zu subtrahieren.

Illegal structure operation *Compiler-Fehler*
Angegebene Operation mit Strukturen nicht zulässig
In C oder C++ können Strukturen nur zusammen mit dem Punkt-Operator (**.**), dem Adreß-Operator (**&**), dem Zuweisungsoperator (**=**) oder als Funktions-Parameter verwendet werden.

Illegal to take address of bit field *Compiler-Fehler*
Adresse eines Bitfeldes darf nicht verwendet werden
Es ist nicht gestattet, die Adresse eines Bitfeldes zu verwenden. Bei anderen Feldern ist dies erlaubt.

Illegal type of entry point *Linker-Fehler*
Nicht-zulässiger Typ eines Einsprungs-Punktes
Es werden nur Einsprungspunkte unterstützt, deren Ziel ein Segment-Index ist

Illegal use of floating point *Compiler-Fehler*
Gleitkommazahlen an dieser Stelle nicht erlaubt

Für Gleitkommazahlen sind folgende Operatoren nicht erlaubt: Schiebeoperatoren (SHL, SHR), Boolesche Operatoren (AND, OR, XOR, NOT), Bedingte Operatoren (**? :**), Indirektion (*) und einige andere. Der Compiler hat einen dieser Fälle gefunden.

Illegal use of member pointer *Compiler-Fehler*

Unzulässiger Gebrauch eines Elementzeigers

Zeiger auf Klassenelemente können nur als Funktionsargument übergeben oder zusammen mit den folgenden Operatoren benützt werden:

- Zuweisung
- Vergleich
- .*
- **->***
- **?:**
- **&&**
- **||**

Sie haben einen Elementzeiger anders verwendet.

Um eine Elementfunktion über einen Zeiger aufzurufen, muß eine Instanz der Klasse existieren.

Illegal use of pointer *Compiler-Fehler*

Zeiger an dieser Stelle nicht erlaubt

Zeiger sind nur im Zusammenhang mit folgenden Operatoren erlaubt:

- Addition (**+**)
- Subtraktion (**-**)
- Zuordnung (**=**)
- Vergleich (**==**)
- Indirektion (*)
- Richtungspfeil (**->>**)

Alle anderen Operationen sind nicht zulässig.

Ill-formed pragma *Compiler-Warnung*

Unbekanntes Pragma

Dieses Pragma entspricht keinem, das der Borland C++-Compiler kennt. Mit der Kommandozeilenoption **-will** können Sie die Meldung kontrollieren. Standardmäßig ist sie eingeschaltet.

Image base address must be a multiple of 0x10000 *Linker-Fehler*

Programmstartadresse muß ein Vielfaches von 0x10000 sein.

Programme müssen an einer 64KByte Grenze beginnen.

Implicit conversion of *Typ1* **to** *Typ2* **not allowed** *Compiler-Fehler*

Implizite Umwandlung von *Typ1* in *Typ2* nicht erlaubt.

Wird eine Elementfunktion über einen Zeiger auf eine abgeleitete Klasse aufgerufen, so muß der Zeiger implizit in einen Zeiger auf die entsprechende Basisklasse umgewandelt werden. Im vorliegenden Fall ist diese implizite Umwandlung nicht erlaubt.

Import record does not match previous definition *Linker-Warnung*

Importrecord stimmt nicht mit vorhergehender Definition überein

(old 32-bit linker) Diese Warnung wird erzeugt, wenn ein IMPDEF Record in einer Importbibliothek zur selben Zeit erscheint, wie der Import von einer DEF-Datei vorgenommen wird. Wenn die Beschreibungen der Importe sich im internen Namen oder im Ordinalwert unterscheiden, wird diese Meldung ausgegeben und die erste Definition wird benutzt. Die Warnung kann über den **-w** Schalter kontrolliert werden.

Improper use of typedef *Bezeichner* *Compiler-Fehler*

typedef *Bezeichner* falsch angewendet

Ihr Quelltext verwendet ein **typedef**-Symbol, wo eine Variable in einem Ausdruck stehen sollte. Überprüfen Sie die Deklaration des Symbols auf mögliche Tippfehler.

Include files nested too deep *Compiler-Fehler*

Include-Dateien zu tief verschachtelt

Der Compiler hat festgestellt, daß die Header-Dateien mehr als 1000 Stufen tief verschachtelt sind. Er nimmt eine Rekursion in den Header-Dateien an und unterbricht die Compilierung mit diesem schweren Fehler.

Dateiname **(***Zeilennummer***): incompatible attribute** *Linker-Fehler*

Dateiname(*Zeilennummer*): Unzulässiges Attribut

TLINK entdeckte nicht-kompatible Segmentattribute in einer CODE- oder DATA-Anweisung. Beispielsweise können PRELOAD und LOADONCALL nicht Attribute für dasselbe Segment sein.

Incompatible type conversion *Compiler-Fehler*

Typ-Umwandlung nicht möglich

Die Typ-Umwandlung konnte nicht durchgeführt werden. Überprüfen Sie die Typen.

Incorrect command-line argument: *Argument* *MAKE-Fehler*

Nichtzulässiges Kommandozeilen-Argument: *Argument*

Sie benutzten nicht zulässige Argumente in der Kommando-Zeile

Incorrect command-line option: *Option* *Compiler-Fehler*

Nicht zulässige Kommando-Zeilen-Option: *Option*

Der Compiler erkannte einen unzulässigen Kommando-Zeilen-Parameter.

Incorrect configuration file option: *Option* *Compiler-Fehler*

Nicht zulässige Option *Option* in der Konfigurationsdatei

Der Compiler erkannte einen unzulässigen Parameter in der Konfigurationsdatei. Prüfen Sie auf einen voranstehenden Bindestrich (**-**).

Incorrect number format *Compiler-Fehler*

Unzulässiges Zahlenformat

Hexadezimalzahlen dürfen keine Nachkommastellen enthalten.

Incorrect use of default *Compiler-Fehler*

abschließender Doppelpunkt **C:**) fehlt

Der Compiler hat in einer **case**-Anweisung keinen Doppelpunkt nach dem Schlüsselwort **default** gefunden.

Incorrect version of RLINK32.DLL *Linker-Fehler*

Nicht zulässige Version von RLINK32.DLL

Die benutzte RLINK32.DLL ist nicht in der korrekten Version vorhanden. Prüfen Sie nach, ob Sie die richtige Version dieser DLL haben.

Initializing enumeration with *Typ* *Compiler-Warnung*

Aufzählung mit dem Typ *Typ* initialisiert

Sie versuchen eine **enum**-Variable mit einem Wert eines anderen Typs zu initialisieren, z.B.

```
enum count {zero, one, two } x = 2;
```

Als Resultat dieser Anweisung wird diese Warnung ausgegeben, weil 2 vom Typ **int** und nicht vom **enum**-Typ **count** ist. Besser ist es, wenn man stattdessen ein Element aus der **enum**-Liste benutzt.

Dies ist ein Fehler, der wie eine Warnung behandelt wird, damit vorhandene Progamme lauffähig bleiben.

Inline assembly not allowed *Compiler-Fehler*

Inline-Assemblierung nicht zulässig

Ihre Quelldatei enthält Inline-Assembleranweisungen, und Sie versuchen Ihr Programm aus der Entwicklungsumgebung heraus zu compilieren. Sie müssen BCC benutzen, um Quelldateien mit Inline-Assembleranweisungen zu compilieren.

Inline assembly not allowed in inline and template functions *Compiler-Fehler*

Assemblerausdrücke in einer Inline- oder einer Template-Funktion nicht gestattet

Der Compiler kann in einer C++ Inline- oder einer Template-Funktion Assemblerausdrücke nicht bearbeiten. Sie können den Inline-Assemblercode löschen oder, wenn es sich um eine Inline-Funktion handelt, daraus ein Makro machen und die Speicherklasse **inline** entfernen.

int and string types compared *MAKE-Fehler*

int- und **string-**Typen werden verglichen

Sie haben versucht, in einer **!if**- oder **!elif**-Anweisung einen **int**-Operanden mit einem string-Operanden zu vergleichen

Internal error - *string* *Linker-Fehler*

Interner Fehler - *string*

Dieser Fehler sollte nicht auftreten. Wenn doch, verständigen Sie die Borland Hotline, und geben Sie den Text der Meldung an.

Internal linker error *Fehlercode* *Linker-Fehler*

Interner Linker-Fehler *Fehlercode*

Ein Fehler im internen Ablauf von TLINK ist aufgetreten. Dieser Fehler sollte nicht auftreten, wird jedoch der Vollständigkeit halber aufgeführt, falls kein weiter spezifizierter Fehler angegeben wird. Falls sich der Fehler nicht beheben läßt, so notieren Sie sich den Fehlercode und setzen Sie sich mit Borland in Verbindung.

Invalid combination of opcode and operands *Compiler-Fehler*

Unzulässige Kombination von Opcode und Operanden

Der integrierte Assembler akzeptiert diese Operandenkombination nicht. Mögliche Gründe dafür sind:

- Es sind zu viele oder zu wenige Operanden für diesen Assembler-Opcode vorhanden.
- Die Anzahl der Operanden stimmt, aber ihre Typen oder ihre Reihenfolge entsprechen nicht dem Opcode, z.B.: **DEC 1**, **MOV AX**, oder **MOV1**, **AX**. Eine Lösung wäre, dem Operanden eine Typen - Bezeichnung voranzustellen, z.B.: **MOV AX**, **WORD PTR foo**.

Invalid entry point offset *Linker-Fehler*

Unzulässiger Einsprungspunkt-Offset

Diese Meldung erscheint nur, wenn Module mit 32-Bit-Datensätzen verkettet werden. Sie bedeutet, daß der Einsprungspunkt-Offset vom Anfangsprogramm die DOS Grenze von 64 kB übersteigt.

Invalid indirection *Compiler-Fehler*

Indirektion nicht zulässig

Der Indirektion-Operator erfordert einen Nicht **void**-Zeiger als Operand

Invalid initial stack offset *Linker-Fehler*

Unzulässiger Startwert für Stack-Offset

Diese Meldung erhalten Sie nur beim Linken von Modulen, die 32-Bit-Records enthalten. Sie bedeutet, daß der Startwert für den Stack-Zeiger die DOS-Beschränkung von 64 kB überschreitet.

Invalid macro argument separator *Compiler-Fehler*

Unzulässiges Trennzeichen zwischen Makro-Argumenten

Die Argumente von Makros müssen grundsätzlich durch Kommas voneinander getrennt sein.Der Compiler hat nach einem Argument ein unzulässiges Zeichen gefunden.

Invalid page size value ignored *Bibliotheken-Fehler*

Unzulässiger Wert für Seitengröße wurde ignoriert

Unzulässige Seitengröße; die Seitengröße muß eine Potenz von 2 sein und darf nicht kleiner als 16 oder größer als 32.768 sein.

Invalid pointer addition *Compiler-Fehler*

Addition zweier Zeiger nicht erlaubt
Es wurde versucht, zwei Zeiger zu addieren.

Invalid register combination (e.g. [BP+BX]) *Compiler-Fehler*

Unzulässige Registerkombination (z.B. [BP+BX])
Der integrierte Assembler hat eine unzulässige Kombination von Registern aufgefunden. Zulässige Index-Registerkombinationen sind [**BX**], [**BP**], [**SI**], [**DI**], [**BX+SI**], [**BX+DI**], [**BP+SI**] und [**BP+DI**]. Andere Index-Registerkombinationen sind nicht erlaubt.

Hinweis: Lokale Variablen (in Prozeduren und Funktionen deklarierte Variablen) befinden sich normalerweise auf dem Stack. Der Zugriff erfolgt über das BP-Register. Der Assembler fügt automatisch [**BP**] in die Referenzen auf derartige Variablen ein. Auch wenn ein Konstrukt wie **local**[**BX**] gültig aussieht (wobei **local** eine lokale Variable ist), ist dies nicht der Fall. Der endgültige Operand würde zu **local**[**BP+BX**].

Invalid segment definition in module *Modul* *Linker-Fehler*

Unzulässige Segmentdefinition im Modul *Modul*
Diese Meldung erhalten Sie grundsätzlich nur dann, wenn ein Compiler eine unsinnige Objektdatei erstellt hat. Wenn dies bei einer mit einem Borland-Compiler erzeugten Datei der Fall ist, compilieren Sie die Datei neu. Falls sich das Problem nicht beheben läßt, setzen Sie sich mit Borland in Verbindung.

Invalid template argument list *Compiler-Fehler*

Unzulässige Parameter in einer Template-Deklaration
In einer Template-Deklaration muß auf das Schlüsselwort **template** eine Liste mit formalen Parametern folgen, die in spitzen Klammern **<<** und **>>** steht. Die Meldung wird ausgegeben, wenn eine ungültige Parameterliste vorhanden ist.

Invalid template qualified name *template::name* *Compiler-Fehler*

Unzulässiger qualifizierter Template-Name *Template::Name*
Bei der Definition eines Template-Klassenelemente müssen die Argumente im Template-Klassennamen, der als linker Operand im **::**-Operator fungiert, mit den formalen Argumenten der Template-Klasse übereinstimmen. Beispiel

```
template <classT> class X
{
   void f();
};
template <classT> void X<T>::f(){}
```

Folgendes ist nicht zulässig:

```
template <classT> void X<int>::f(){}
```

Invalid use of dot *Compiler-Fehler*

Unzulässige Verwendung des Operators Punkt (**.**).
Auf den Operator Punkt (**.**) muß unmittelbar der Bezeichner folgen.

Invalid use of template *Template* *Compiler-Fehler*

Unzulässige Verwendung des Template *Template*
Außerhalb einer Template-Definition ist die Verwendung eines Template-Klassennamens ohne Argumente nicht zulässig. So können Sie beispielsweise **vector**<**int**>, aber nicht **vector** angeben.

Irreducible expression tree *Compiler-Fehler*

Interner Compiler-Fehler bei der Auswertung eines Ausdrucks

Hier handelt es sich um einen Compilerfehler. Ein Ausdruck in der angegebenen Zeile verhindert, daß der Code-Generator Code erzeugen kann. Lassen Sie den Ausdruck weg. Sollten Sie jemals diesen Fehler erhalten, setzen Sie sich bitte mit Borland in Verbindung.

Basis **is an indirect virtual base class of** *Klasse* *Compiler-Fehler*

Basis ist eine indirekte virtuelle Basisklasse von *Klasse*

Ein Zeiger auf ein C++ Element einer virtuellen Basisklasse kann nicht angelegt werden. Es wurde versucht, einen derartigen Zeiger zu erstellen (entweder direkt oder über eine Typen-Umwandlung).

Bezeichner **is assigned a value that is never used** *Compiler-Warnung*

Bezeichner wird ein Wert zugewiesen, der nie verwendet wird

Die Variable erscheint in einer Zuweisung, wird aber nirgends in der gerade beendeten Funktion verwendet. Die Warnung wird erst ausgegeben, wenn der Compiler eine rechte geschweifte Klammer findet.

Identifier is declared as both external and static. *Compiler-Warnung*

Bezeichner ist sowohl **extern** als auch **static** deklariert

Der Bezeichner trat in einer Deklaration auf, die ihn implizit oder explizit als global oder **extern** kennzeichnet, aber auch in einer **static** Deklaration. Überprüfen Sie alle Deklarationen für diesen Bezeichner.

Bezeichner **is declared but never used** *Compiler-Warnung*

Bezeichner ist deklariert, wird aber nicht verwendet

In Ihrer Quelltextdatei ist die angegebene Variable als Teil des gerade endenden Blocks deklariert, sie wurde aber nie verwendet. Die Warnung wird ausgegeben, wenn der Compiler auf die rechte geschweifte Klammer der Blockanweisung oder Funktion trifft. Die Variable wird am Beginn der Blockanweisung oder Funktion deklariert.

Symbol is duplicated in module *Modul* *Linker-Fehler*

Symbol in Modul *Modul* dupliziert

Ein Konflikt zwischen zwei Symbolen, die in einem Modul definiert worden sind. Der Fehler erscheint, wenn beide in einer OBJ-Datei auftreten, oder eine Warnung, wenn TLINK beide Symbole in einer Bibliothek findet. In diesem Fall wird TLINK die erste Definition verwenden.

constructor is not a base class of *Klasse* *Compiler-Fehler*

Konstruktor ist keine Basisklasse von Klasse

Ein C++ Klassenkonstruktor Klasse versucht einen Basisklassenkonstruktor Konstruktor aufzurufen, oder Sie versuchen die Zugriffsrechte von Klasse::Konstruktor zu verändern. Konstruktor ist keine Basisklasse von Klasse. Bitte überprüfen Sie die Deklaration.

Bezeichner **is not a member of** *Struktur* *Compiler-Fehler*

Bezeichner ist kein Element von *Struktur*

Sie versuchen fälschlicherweise, einen Bezeichner als Element einer Struktur zu behandeln, zu der er nicht gehört. Prüfen Sie die Deklaration nach.

Bezeichner **is not a non-static data member and can't be initialized here** *Compiler-Fehler*

Bezeichner ist kein nicht-statisches Datenelement und kann hier nicht initialisiert werden

Nur Datenelemente können mit dem Initialisierer in einem Konstruktor initialisiert werden, hier lag aber ein statisches oder ein Funktionselement vor.

Bezeichner **is not a parameter** *Compiler-Fehler*

Bezeichner ist kein Parameter

Im Abschnitt Parameterdeklarationen einer Funktionsdefinition alten Stils wird der Bezeichner deklariert, aber nicht als Parameter aufgelistet. Entfernen Sie die Deklaration oder ergänzen Sie *Bezeichner* als Parameter.

Typ **is not a polymorphic class type** *Compiler-Fehler*
Typ ist kein polymorpher Klassentyp
Ein dynamic_cast wurde mit einem Zeiger auf eine Klasse benutzt, die mit deaktivierter -RT-Option compiliert wurde.

Bezeichner **is not a public base class of** *Klassentyp* *Compiler-Fehler*
Bezeichner ist keine öffentliche Basisklasse von *Klassentyp*
Der rechte Operand von einem **.*** , **->>*** oder **::operator** ist kein Zeiger auf ein Element einer Klasse, die entweder eine eindeutige zugreifbare Basisklasse des linken Operanden des Klassentyps oder mit einer derartigen Klasse identisch ist.

Dateiname **is not a valid library** *Linker-Warnung*
Dateiname ist keine gültige Bibliothek
Der Fehler tritt auf, wenn eine Datei mit einer Bibliothekssektion, die kein gültiges Modul enthält, verkettet wurde .

Element **is not accessible** *Compiler-Fehler*
Element nicht zugänglich
Sie versuchen, auf ein Element einer C++ Klasse zuzugreifen, das **private** oder **protected** und von dieser Funktion nicht referenzierbar ist. Dies geschieht manchmal, wenn eine zugängliche, überladene Elementfunktion (oder ein Konstruktor) aufgerufen werden soll, die Argumente jedoch einer unzugänglichen Funktion entsprechen. Mögliches Überladen wird immer vor den Zugriffsrechten geprüft. Wandeln Sie einen oder mehrere Parameter um, um die gewünschte zugreifbare Funktion auszuwählen.

Last parameter of *Operator* **must have type int** *Compiler-Fehler*
Der letzte Parameter von *Operator* muß vom Typ **int** sein
Wenn ein Suffix-Operator **operator++** oder **operator--** deklariert wird, muß der letzte Parameter vom Typ int sein

Library contains COMDEF records - extended dictionary not created *Bibliotheken-Warnung*
Bibliothek enthält COMDEF-Records - erweitertes Verzeichnis nicht erzeugt
Wenn die Meldung Library contains COMDEF records angezeigt wird, enthält ein Objekt-Record, der der Bibliothek hinzugefügt werden soll, einen COMDEF-Record. Dies verträgt sich nicht mit der Option für ein erweitertes Verzeichnis.

Library too large, restart with library page size *Größe* *Bibliotheken-Fehler*
Bibliothek zu groß, erneuter Start mit Bibliothek-Seitengröße *Größe*
Die Bibliothek konnte nicht mit der bestehenden Bibliotheks-Seitengröße erzeugt werden.

Linkage specification not allowed *Compiler-Fehler*
Bindungsspezifikation nicht zulässig
Bindungsspezifikation wie **extern** »*C«* sind nur auf Dateiebene erlaubt. Plazieren Sie diese Funktionsdeklaration auf Dateiebene.

Linker name conflict for *Funktion* *Compiler-Fehler*
Linker Namenskonflikt für *Funktion*
Wenn der Name einer C++ Inline-Funktion zu lang ist und deswegen gekürzt werden muß, (meist geschieht dies mit Templates) und dieser abgeschnittene Name dem Namen einer vorher generierten Funktion gleicht, erscheint eine Compiler-Fehlermeldung. Sie beheben den Fehler entweder durch Änderung des Namens oder einer Kompilierung mit der Option **-Vs**.

Linker stack overflow *Linker-Fehler*
Überlauf des Linker-Stacks
Der Linker benutzt eine rekursive Prozedur, um Module die in ein ausführbares Duplikat einer Bibliothek eingefügt werden müssen zu markieren. Im Extremfall kann diese Prozudur einen Stack-Overflow erzeugen. Wenn diese Fehlermeldung erscheint, entfernen Sie einige Module aus Bibliotheken, verbinden Sie diese mit den Objektdateien im Link-Vorgang und starten erneut.

Lvalue required *Compiler-Fehler*
L-Wert erforderlich

Die linke Seite eines Zuweisungsoperators muß ein adressierbarer Ausdruck sein. Dies schließt numerische und Zeigervariablen, Referenzen auf Strukturfelder, Indirektionen durch Zeiger oder ein indiziertes Array-Element ein.

Macro argument syntax error *Compiler-Fehler*
Makro Argument Syntax Fehler
Ein Argument in einer Makrodefinition muß ein Bezeichner sein. Der Compiler hat für Bezeichner ungültige Zeichen angetroffen, obwohl an diesem Punkt ein Argument erwartet wurde.

Macro expansion too long *Compiler-Fehler*
Makro-Erweiterung zu lang
Ein Makro kann nicht mehr als 4096 Zeichen erweitert werden.

Macro expansion too long *MAKE-Fehler*
Makro Erweiterung zu lang
Ein Makro kann nicht zu mehr als 4096 Zeichen erweitert werden. Dieser Fehler tritt oft auf wenn sich ein Makro rekursiv ausweitet.

Macro substitute text *Zeichenkette* **is too long** *MAKE-Fehler*
Makro Ersatz-*Zeichenkette* ist zu lang
Macro replace text *Zeichenkette* **is too long** *MAKE-Fehler*
Makro Auswechsel-*Zeichenkette* ist zu lang
Der Makro-Ersatz oder die Auswechsel-Zeichenkette übersteigt den internen Puffer von 512 Bytes des MAKE.

main must have a return type of int *Compiler-Fehler*
main muß einen Integerwert liefern
Die Funktion **main** muß bestimmte Anforderungen erfüllen. Dazu gehört, daß **main** nur mit einem Rückgabewert vom Typ **int** deklariert sein darf.

Malformed command-line *Compiler-Fehler*
Falsche Befehls-Zeile
Ein unzulässiger Eintrag in der Befehlszeile wurde gefunden.

Matching base class function for *Funktion* **has different dispatch number** *Compiler-Fehler*
Die entsprechende Basisklassenfunktion für *Funktion* hat eine andere Zuordnungsnummer
Wenn eine DDVT-Funktion in einer abgeleiteten Klasse deklariert ist, muß die entsprechende Funktion der Basisklasse diesselbe Zuordnungsnummer wie die abgeleitete Funktion haben.

Matching base class function for *Funktion* **is not dynamic** *Compiler-Fehler*
Die entsprechende Basisklassenfunktion für *Funktion* ist nicht dynamisch
Wenn eine DDVT-Funktion in einer abgeleiteten Klasse deklariert ist, muß die entsprechende Funktion der Basisklasse dynamisch sein.

Maximum precision used for member pointer type *Typ* *Compiler-Warnung*
Höchste Genauigkeit für Zeigertyp *Typ* auf ein Klassenelement verwendet
Wenn Sie einen Zeigertyp auf ein Klassenelement verwenden, die Klasse bereits vollständig definiert wurde und die **-Vmd**-Option benutzt wird, muß der Compiler die allgemeine (und am wenigsten effiziente) Repräsentation für diesen Elementzeigertyp anwenden. Dieses erzeugt einen wenig effizienten Code (und benötigt einige Byte zur Darstellung des Zeigers) und kann Probleme hervorrufen, wenn Sie separat compilieren.

Member function must be called or its address taken *Compiler-Fehler*
Die Elementfunktion muß aufgerufen oder ihre Adresse ermittelt werden
Wenn in einem Ausdruck eine Elementfunktion verwendet wird, muß diese aufgerufen oder ihre Adresse mit dem Operator **&** ermittelt werden. In diesem Fall wurde eine Elementfunktion in einem unzulässigen Kontext eingesetzt.

Member identifier expected *Compiler-Fehler*

Es wird ein Element-Bezeichner erwartet

Der Name einer Struktur oder eines Klassenelementes wurde erwartet, aber nicht gefunden. Die rechte Seite eines Punkt- oder Pfeil- Operators muß der Name eines Elements in der Struktur oder der Name einer Klasse auf der linken Seite des Operators sein.

Member is ambiguous: *member1* **and** *member2* *Compiler-Fehler*

Element ist mehrdeutig

Sie müssen die Elementreferenz mit dem entsprechenden Basisklassennamen referenzieren. In der C++ Klasse *Klasse* kann das Element *Element* in mehreren Basisklassen ohne genaue Zuordnung vorhanden sein. Dies ist nur bei Mehrfachvererbung der Fall, in denen der Elementname in jeder Basisklasse nicht durch denselben Elementnamen in einer abgeleiteten Klasse mit demselben Pfad verborgen wird. Die Regeln erfordern, daß dieser Test auf Mehrdeutigkeit vor der Überprüfung der Zugriffsrechte(**private**, **protected**, **public**) durchgeführt wird. Aus diesem Grund können Sie diese Meldung auch erhalten, wenn nur auf ein (oder kein) Element zugegriffen werden kann.

Member *Element* **cannot be used without an object** *Compiler-Fehler*

Element ohne Objekt verwendet

Hier haben Sie *Klasse::Element* angegeben, wobei *Element* ein gewöhnliches (nicht statisches) Element ist und keiner Klasse angehört.Es gibt keine Klasse, die mit diesem Element verknüpft werden kann.
obj.Klasse::Element ist zulässig, *Klasse ::Element* jedoch nicht

Member *Element* **has the same name as its class** *Compiler-Fehler*

Element *Element* hat denselben Namen wie seine Klasse

Ein statisches Datenelement, ein Aufzählungstyp, ein Element einer anonymen Variante oder ein geschachtelter Typ dürfen nicht denselben Namen haben wie ihre Klasse. Nur Elementfunktionen oder nicht statische Elemente können denselben Namen wie ihre Klasse tragen

Member *Element* **is initialized more than once** *Compiler-Fehler*

Element *Element* mehrmals initialisiert

In einem C++ Klassenkonstrukor enthält die Initialisierungsliste nach dem Konstruktorkopf denselben Elementennamen mehrmals

Member pointer required on right side of .* or ->>* *Compiler-Fehler*

Auf der rechten Seite der Operatoren **.***oder **->>*** muß ein Elementzeiger stehen

Die rechte Seite der Operatoren **.***oder **->>*** muß als Zeiger auf ein Element der Klasse deklariert sein, die auf der linken Seite des Operators angegeben ist. Bei einer Fehlermeldung ist die rechte Seite kein Elementzeiger

Memory full listing truncated *Bibliothek Warnung*

Speicher voll, Listendatei unvollständig

Die Bibliotehksverwaltung hatte bei der Erstellung einer Bibliotheks-Listendatei nicht mehr genügend Speicher zur Verfügung. Es wird zwar eine Listendatei erzeugt, diese ist jedoch unvollständig.

Memory reference expected *Compiler-Fehler*

Speicher Referenz erwartet

Der integrierte Assembler benötigt eine Speicherreferenz. Ein häufiger Grund für diesen Fehler liegt darin, daß die eckigen Klammern um den Operanden eines Index-Registers fehlen,
z.B. MOV AX, BX+SI anstelle von MOV AX,[BX+SI]

Misplaced break *Compiler-Fehler*

Falsch positionierte **break** Anweisung

Der Compiler hat eine **break** Anweisung außerhalb einer **switch** Anweisung oder Schleife gefunden

Misplaced continue *Compiler-Fehler*

Falsch positionierte **continue** Anweisung

Der Compiler hat eine **continue** Anweisung außerhalb einer **switch** Anweisung oder Schleife gefunden

Misplaced decimal point *Compiler-Fehler*

Falsch positionierter Dezimalpunkt
Der Compiler hat im Exponenten von Gleitkommazahlen einen Dezimalpunkt gefunden

Misplaced elif directive *Compiler-Fehler*

Falsch positionierte **elif** Anweisung
Der Compiler hat eine **elif** Anweisung ohne vorangehende **if**, **ifdef** oder **ifndef**-Anweisung gefunden.

Misplaced elif statement *MAKE-Fehler*

Falsch positionierte **elif** Aussage
Einer **elif** Anweisung fehlt eine passende **if** Anweisung

Misplaced else *Compiler-Fehler*

Falsch positionierte **else** Anweisung
Der Compiler hat eine **else** Anweisung ohne dazugehöriges **if** gefunden. Ein überzähliges **else** kann zu dieser Meldung geführt haben, aber auch ein überzähliger Strichpunkt, geschweifte Klammern oder ein Syntaxfehler

Misplaced else directive *Compiler-Fehler*

Falsch positionierte **else** Anweisung
Der Compiler hat eine **else** Anweisung ohne vorangehendes **if**, **ifdef**, oder **ifndef** gefunden

Misplaced else statement *MAKE-Fehler*

Falsch positionierte **else** Aussage
Einer **else** Anweisung fehlt eine passende **if** Anweisung

Misplaced endif directive *Compiler-Fehler*

Falsch positionierte **endif** Anweisungen
Der Compiler hat eine **endif** Anweisung ohne vorangehendes **if**, **ifdef** oder **ifndef** gefunden

Misplaced endif statement *MAKE-Fehler*

Falsch positionierte **endif** Aussage
Einer **endif** Anweisung fehlt eine passende **if** Anweisung

Filename(Line number):missing internal name *Linker-Fehler*

Dateiname(*Zeilennummer*): Interner Name fehlt
Im IMPORTS-Bereich der Moduldefinitionsdatei trat eine durch Modulname und Ordinalwert spezifizierte Referenz auf einen Einsprung auf.Wird ein Einsprung durch den Ordinalwert bestimmt, so muß dieser Importdefinition ein interner Name zugewiesen werden. Über diesen internen Namen bezieht sich ihr Programm auf die Importdefinition. Die in der Moduldefinitionsdatei zu verwendende Syntax lautet:

```
<InternerName>=<ModulName>.<Einsprung>
```

Mixed common types in module ***Modul*****. Cannot mix COMDEFs and VIRDEFs** *Linker-Fehler*

Unterschiedliche Standardtypen in Modul *Modul*. COMDEFs und VIRDEFs dürfen nicht gemischt werden
Sie dürfen COMDEFs und VIRDEFs nicht mischen. Schalten Sie den **-FC** Schalter aus, um die Generierung von COMDEFs zu vermeiden oder den **-Vs** Schalter, um die Erstellung von VIRDEFs zu verhindern.

Mixing pointers to different char types *Compiler-Warnung*

Vermischen von Zeigern auf **signed** und **unsigned char**
Sie führten die Konvertierung von einem Zeiger auf **signed** in **unsigned char** oder umgekehrt ohne Angabe einer expliziten Typumwandlung durch. (Genau genommen ist dies nicht zulässig, oft jedoch ohne Bedeutung.)

Multiple base classes require explicit class names *Compiler-Fehler*

Mehrfache Basisklassen erfordern explizite Klassennamen

In einem Klassenkonstruktor in C++ muß jeder Aufruf des Klassenkonstruktors der Basisklasse im Konstruktorkopf den Namen der Basisklasse enthalten, wenn mehr als eine unmittelbare Basisklasse existiert.

Multiple declaration for *Bezeichner* — *Compiler-Fehler*

Bezeichner mehrfach deklariert

Der Bezeichner wurde fälschlicherweise mehrfach deklariert. Dies kann aus widersprüchlichen Angaben herrühren wie: **int** a; **double** a; einer zweimal und verschieden deklarierten Funktion, einer wiederholten Sprungadresse in derselben Funktion oder einer wiederholten Deklaration außer einer **extern** Funktion oder einer einfachen Variablen.

Multiple entry points defined — *Linker-Fehler*

Mehrere Einsprungpunkte definiert

Für die Anwendung wurden mehrere Einsprungpunkte definiert. Dieses kann auftreten, wenn Sie den Startup-Code zweimal angeben, oder wenn Sie Assembler-Code benutzen, der eine Startadresse für die Anwendung definiert.

Multiple stack segments found. The most recent one will be used — *Linker-Warnung*

Mehrere Stacksegmente gefunden. Das letzte Segment wird benutzt

Sind zwei Stacksegmente mit unterschiedlichen Namen in den Object-Modulen definiert, wird diese Warnung ausgegeben. Der Startup-Code definiert ein Stacksegment für die Anwendung. Mit dem **/w**-Schalter der TLINKOptions kann die Warnung kontrolliert werden.

Operator **must be a member function** — *Compiler-Fehler*

Operator muß eine Elementfunktion sein

Die meisten Operatorfunktionen von C++ können Elemente von Klassen oder gewöhnliche Nicht-Elementfunktionen sein. Die folgenden müssen aber Elemente von Klassen sein: **operator =**, **operator ->>**, **operator ()** und Typumwandlungen. Diese Operatorfunktionen müssen eine Elementfunktion sein.

*Operato***must be a member function or have a parameter of class type** — *Compiler-Fehler*

Operator muß Elementfunktion sein oder ein Element des Typs **class** enthalten

Die meisten C++ Operatorfunktionen müssen einen expliziten oder impliziten Parameter des Typs **class** aufweisen. Diese Operatorfunktion wurde außerhalb einer Klasse deklariert und enthält keinen expliziten Parameter vom Typ **class**.

Operator **must be a previously defined class or struct** — *Compiler-Fehler*

Operator muß vorher als **class** oder **struct** definiert werden

Sie versuchen, *Bezeichner* als Basisklasse zu definieren, aber entweder ist er keine Klasse oder nicht vollständig definiert. Ändern Sie den Namen oder die Deklarationen.

Operator **must be a previously defined enumeration tag** — *Compiler-Fehler*

Operator muß vorher als **enum**-tag definiert werden

Sie versuchen *Bezeichner* als Teil eines **enum** anzusprechen, er wurde aber nicht so deklariert. Ändern Sie den Namen oder die Deklarationen.

Operator **must be declared with no parameters** — *Compiler-Fehler*

Operator muß ohne Parameter deklariert werden

Diese C++ Operatorfunktion ist fälschlicherweise mit Parametern deklariert

Operator **must be declared with one parameter** — *Compiler-Fehler*

Operator muß mit einem Parameter deklariert werden

Diese C++ Operatorfunktion ist fälschlicherweise mit mehr als einem Parameter deklariert

Operator **must be declared with one or no parameters** — *Compiler-Fehler*

Operator muß mit einem oder keinem Parameter deklariert werden

Wenn **operator ++** oder **operator --** als Elementfunktion deklariert werden, müssen sie entweder keine Parameter (in der prefix-Version), oder einen Parameter vom Typ **int** (in der postfix-Version) übernehmen.

Operator **must be declared with one or two parameters** *Compiler-Fehler*

Operator muß mit einem oder zwei Parametern deklariert werden

Wenn **operator ++** oder **operator --** als Nicht-Elementfunktion deklariert werden, müssen sie entweder einen Parameter (in der prefix-Version), oder zwei Parameter (in der postfix-Version) übernehmen.

Operator **must be declared with two parameters** *Compiler-Fehler*

Operator muß mit zwei Parametern deklariert werden

Diese C++ Operatorfunktion ist fälschlicherweise nicht mit zwei Parametern deklariert

Must take address of a memory location *Compiler-Fehler*

Keine Adresse für Speicherposition vorhanden

In der Quelltextdatei ist ein Adreßoperator mit einem unzulässigen Ausdruck vorhanden, z.B. einer Registervariablen

Need an identifier to declare *Compiler-Fehler*

Bezeichner für Deklaration nötig

In diesem Zusammenhang wird ein Bezeichner erwartet, um die Deklaration zu vervollständigen. Es kann z.B. ein **typedef** ohne Namen oder ein zusätzlicher Strichpunkt auf Dateiebene vorliegen. In C++ kann das ein Klassenname sein, der fälschlich für eine anderere Bezeichnerart verwendet wird.

No: following the ? *Compiler-Fehler*

Kein Doppelpunkt nach dem Fragezeichen

In diesem Ausdruck sind die Operatoren **?** und **:** falsch plaziert. Vielleicht fehlt der Doppelpunkt, oder Klammern sind falsch verschachtelt oder fehlen.

No base class to initialize *Compiler-Fehler*

Basisklasse fehlt

Dieser C++ Klassenkonstruktor versucht implizit den Klassenkonstruktor einer Basisklasse aufzurufen. Für diese Klasse wurde aber keine Basisklasse deklariert. Überprüfen Sie Ihre Deklarationen.

No closing quote *MAKE-Fehler*

Keine erforderliche, schließende spitze Klammer vorhanden

Keine erforderliche, schließende spitze Klammer für ein Zeichenausdruck in einem **!if** oder **!elif-**Ausdruck vorhanden

No declaration for function *Funktion* *Compiler-Warnung*

Keine Deklaration für Funktion *Funktion*

Diese Warnung erhalten Sie, wenn Sie eine Funktion aufrufen, ohne diese vorher deklariert zu haben. In C kann eine Funktion ohne Prototyp deklariert werden z.B. **int** func();In C++ stellt jede Funktionsdefinition auch einen Prototyp dar; das Beispiel ist äquivalent zu **int** func(**void**); Die Deklaration kann hier im klassischen oder modernen (Prototyp) Stil erfolgen.

No .DEF file; using defaults *Linker-Warnung*

Keine Moduldefinitionsdatei angegeben; Verwendung der Voreinstellung

TLINK wurde mit einer Windows-Option aufgerufen, es wurde aber keine Moduldefinitionsdatei angegeben. Sie können das Erscheinen der Warnung mit dem **-w**-Schalter der TLINK-Optionen kontrollieren.

No file name ending *Compiler-Fehler*

Dateiname nicht korrekt abgeschlossen

Hinter dem Dateinamen in einer **#include**-Anweisung fehlt die erforderliche, schließende spitze Klammer oder das abschließende Anführungszeichen.

No filename ending *MAKE-Fehler*

Dateiname nicht korrekt abgeschlossen

Dem Dateinamen in einer **include** Anweisung fehlt die erforderliche schließende spitze Klammer oder das abschließende Anführungszeichen

No file names given *Compiler-Fehler*

Keinen Dateinamen angegeben.
In der Kommandozeile des Compilers (BCC) sind keine Dateinamen vorhanden. Sie müssen aber einen Quelltextdateinamen angeben.

No internal name for IMPORT in .DEF file *Bibliotheken-Fehler*

Keinen internen Namen gewählt, um in eine DEF-Datei zu importieren.
Die DEF-Datei enthält einen semantischen Fehler. Der Anwender hat wahrscheinlich vergessen, einen internen Namen für einen Import vor den Modulnamen zu setzen.

Ein Beispiel:

```
IMPORTS
   _foo.1
```

Hier ist *_foo* die Funktion, die importiert werden soll, aber die richtige Syntax lautet:

```
IMPORTS
   _foo=mydll.1
```

No macro before = *MAKE-Fehler*

Namenszuweisung vor einem **=**
Sie müssen dem Makro einen Namen geben, bevor Sie es einem Wert zuweisen.

No match found for wildcard expression *MAKE-Fehler*

Keinen passenden Ausdruck gefunden
Es gibt keine Dateien, die diesem Ausdruck gleichen. Ein Beispiel:

```
prog.exe: *.obj
```

MAKE schickt diese Fehlermeldung, wenn keine Dateien mit Endung OBJ im aktuellen Verzeichnis vorhanden sind.

No output file specified *Linker-Fehler*

Keine Ausgabedatei angegeben
Sie haben keine Ausgabedatei spezifiziert. Da der Linker vorgabemäßig auf den ersten .OBJ-Namen eingestellt ist, wird dieser Fehler meist dadurch verursacht, daß keine .OBJ-Dateien angegeben wurden.

No program entry point *Linker-Warnung*

Keine Startadresse für Programm definiert
Diese Warnung bedeutet, daß kein Modul vorhanden ist, in dem die Startadresse für das Programm angegeben ist. Sie haben wahrscheinlich vergessen, das Initialisierungsmodul C0x.OBJ zu linken. Die Warnung kann über den **-w**-Schalter der TLINK-Optionen ein- bzw. ausgeschaltet werden.

No program starting address defined *Linker-Warnung*

Keine Startadresse für Programm definert
Diese Warnung bedeutet, daß kein Modul vorhanden ist, in dem die Startadresse für das Programm angegeben ist. Sie haben wahrscheinlich vergessen, das Initialisierungsmodul C0x.OBJ zu linken.

No stack *Linker-Warnung*

Kein Stack
Diese Warnung wird ausgegeben, wenn in keiner Objektdatei und keiner Bibliothek ein Stacksegment definiert ist. Außer beim Erstellen von DLLs, erzeugt dieses einen Fehler.

Wenn C++ für für ein Speichermodell außer tiny diese Meldung ausgibt, prüfen Sie ob Sie die richtige COx Start-Datei benutzen.

No stub for fixup at ***Segment:xxxxh*** **in module** ***Modul*** *Linker-Warnung*

Kein Stub für Fixup bei *Segment:xxxxh* im Modul *Modul*

Sie erhalten diese Warnung, wenn das Ziel eines Fixup ein Overlay ist, aber kein Stub für ein externes Ziel vorhanden ist. Das ist normalerweise der Fall, wenn ein Symbol in einem Overlay, das von demselben Modul referenziert wird, nicht als **public** deklariert ist.

No terminator specified for in-line file operator *MAKE-Fehler*

Keine Endung für ein in-line Datei Operator deklariert
Die MAKE Datei enthält entweder **&&** oder **<<** -Befehlszeilen-Operatoren um eine in-line Datei zu starten, doch die Datei ist nicht beendet.

Non-const function *Funktion* **called for const object** *Compiler-Warnung*

Nicht konstant Funktion *Funktion* für konstantes Objekt aufgerufen
Eine nicht konstante Elementfunktion wurde für ein Objekt vom Typ **const** aufgerufen. Dies ist ein Fehler, der aber auf eine Warnung herabgestuft wurde, damit schon existierende Programme compiliert werden können.

Nonportable pointer comparison *Compiler-Warnung*

Nicht portierbarer Zeigervergleich. Ihr Quelltext enthält einen Vergleich von einem Zeiger mit einem Nicht-Zeiger der nicht NULL ist.
Verwenden Sie eine Umwandlung, wenn Sie diese Warnung unterdrücken wollen und der Vergleich korrekt ist. Mit der Kommandozeilenoption **-wcpt** lassen Sie die Meldung anzeigen oder unterdrücken sie. Standardmäßig wird die Warnung angezeigt.

Nonportable pointer conversion *Compiler-Warnung*

Nicht portierbare Zeigerumwandlung
Ein von Null verschiedener, ganzzahliger Wert wird in einem Zusammenhang verwendet, in dem ein Zeiger oder ein ganzzahliger Wert erforderlich ist; Größe von Ganzzahl-Typ und Zeiger sind gleich. Verwenden Sie eine explizite Umwandlung, wenn Sie dies wirklich wollen. Mit der Kommandozeilenoption **-wrpt** können Sie die Meldung kontrollieren. Standardmäßig ist sie eingeschaltet.

Nonportable pointer conversion *Compiler-Fehler*

Typfehler bei Zeigerumwandlung
Es ist hier eine implizite Umwandlung von einem Zeiger in einen Integerwert verlangt, die Typen haben aber nicht diesselbe Größe. Nur eine explizite Typenumwandlung ist möglich. Die Umwandlung ist eventuell unsinnig, noch einmal überprüfen ob gewünschter Ablauf.

Nonresident Name Table is greater than 64 kB *Linker-Warnung*

nicht-residente Namenstabelle überschreitet 64 kB
Der Linker arbeitet weiter und ignoriert alle folgenden nicht-residenten Namenstabellen, die er während der Verkettung antrifft.

Nontype template argument must be of scalar type *Compiler-Fehler*

Ein nicht typisiertes Template-Argument muß ein skalarer Typ sein
Ein formales nicht typisiertes Template-Argument muß ein skalarer Typ sein. Es kann vom Typ **int**, **enum** oder ein Zeigertyp sein

Non-ANSI Keyword used: *Schlüsselwort* *Compiler-Fehler*

Kein ANSI-Schlüsselwort: *Schlüsselwort*
Ein Nicht-ANSI Schlüsselwort (z.B.: »__fastcall«) wurde benutzt, wobei strikte ANSI-Konformität über die **-A**-Option eingestellt war.

Non virtual function *Funktion* **declared pure** *Compiler-Fehler*

Nicht-virtuelle Funktion *Funktion* als abstrakt erklärt
Nur virtuelle Funktionen können als abstrakte Funktionen deklariert werden, damit abgeleitete Klassen sie überschreiben können.

Non volatile function *Funktion* **called for volatile object** *Compiler-Warnung*

Eine nicht-volatile Funktion *Funktion* wurde für ein Objekt vom Typ volatile aufgerufen
Nur explizit als **volatile** deklarierte Elementfunktionen können mit Objekten vom Typ **volatile** aufgerufen werden.

Dateiname **not a MAKE** *MAKE-Fehler*

Dateiname nicht zulässig für MAKE
Die Datei, die Sie mit der Option **-f** gewählt haben ist keine MAKE-Datei.

Not an allowed type *Compiler-Fehler*

Typ ist nicht zulässig
Sie haben einen unzulässigen Typ deklariert, beispielsweise eine Funktion, die eine Funktion oder einen Array zurückliefert.

Not enough memory *MAKE-Fehler*

Nicht genügend Speicher vorhanden
Der komplette Arbeitsspeicher ist ausgeschöpft.

Not enough memory *Linker-Fehler*

Nicht genügend Speicherplatz
Es ist nicht genügend Speicher für den Betrieb des TLINK vorhanden. Versuchen Sie eine derzeit geladene TSR-Anwendung zu entfernen oder verringern Sie den Umfang momentan aktiver RAM-Disks.Im Real-Modus versuchen Sie es mit der Option **-S**. Starten Sie dann TLINK erneut.

Not enough memory for command-line buffer *Bibliotheken-Fehler*

Nicht genügend Speicherplatz für Befehlszeilen-Puffer
Dieser Fehler tritt auf wenn die Bibliothek keinen Speicher zur Verfügung hat.

Module **not found in library** *Bibliotheken-Warnung*

Modul nicht gefunden
Entweder **_** oder ***** wurde in einer Bibliothek versucht auszuführen und das Objekt existiert nicht.

Null pointer assignment *Laufzeit-Fehler*

Null Zeiger Zuweisung
Nach Beendigung eines Small oder Medium Programms wird automatisch überprüft, ob der Inhalt der ersten Bytes im Datensegment des Programms geändert wurde. Bei einem richtig funktionierenden Programm würde das nicht passieren. Diese Meldung informiert Sie darüber, daß höchstwahrscheinlich ein Wert über einen nicht-initialisierten Zeiger zugewiesen wurde. Selbst wenn das Programm voll funktionstüchtig erscheint, sollten Sie diese Meldung als Hinweis auf einen ernsthaften Fehler behandeln. Unterlassen Sie nämlich die Korrektur dieses Fehlers, ist das Verhalten des Programms nicht vorhersehbar und könnte gelegentlich sogar zu einem Systemabsturz(bei den Speichermodellen Large, Compact und Huge) führen. Benutzen Sie den integrierten Debugger, um die Fehlerstelle zu ermitteln.

Numeric constant too large *Compiler-Fehler*

Numerische Konstante zu groß
Es können keine String- oder Zeichen- Escape-Sequenzen erzeugt werden, die größer als hex \xFF oder oktal \377 sind. Zwei-Byte-Zeichenkonstanten können mit Hilfe eines zweiten Backslash angegeben werden. Zum Beispiel steht \x0D\x0A für eine Zwei-Byte-Konstante. Ein numerisches Literal direkt nach einer Escape-Sequenz sollte so abgetrennt werden:

```
printf("\x0D" "12345");
```

Damit wird ein Zeilenvorschub und danach 12345 ausgegeben.

Object module *Dateiname* **is invalid** *Bibliotheken-Warnung*

Ungültiges Objekt-Modul *Dateiname*
TLIB konnte den, der Bibliothek hinzuzufügenden, Header-Record des Objekt-Moduls nicht interpretieren. TLIB geht davon aus, daß es sich um ein unzulässiges Modul handelt.

Objects of type *Typ* cannot be initialized with {} *Compiler-Fehler*

Objekte vom Typ *Typ* können nicht mit **{}** initialisiert werden

Gewöhnliche Strukturen in C können mit einer Anzahl von Werten in geschweiften Klammern initialisiert werden. Klassen können in C++ nur durch Konstruktoren initialisiert werden, wenn die Klasse Konstruktoren, private Elemente, Funktionen oder Basisklassen enthält, die virtuell sind.

Old debug information in module *Modul* will be ignored *Linker-Warnung*

Alte Debug-Informationen im Modul *Modul* werden ignoriert, da sie inkompatibel zum Linker sind.

Debug-Informationen innerhalb der OBJ-Dateien, die zu TLINK nicht kompatibel sind werden ignoriert.

Only <<KEEP or <<NOKEEP *MAKE-Fehler*

Nur auf KEEP oder NOKEEP beschränken

Sie haben zusätzlich etwas zu KEEP oder NOKEEP deklariert.

Only member functions may be const or volatile *Compiler-Fehler*

Nur Elementfunktionen können vom Typ **const** oder **volatile** sein

Etwas anderes als eine Elementfunktion wurde als **const** oder/und **volatile** deklariert.

Only one of a set of overloaded functions can be »C« *Compiler-Fehler*

Nur eine der überladenen Funktionen darf mit »C« deklariert werden

Alle C++ Funktionen werden per Voreinstellung überladen, wobei der Compiler jeder Funktion einen neuen Namen zuweist. Wollen Sie diese Umbenennung verhindern, indem Sie die Funktion als **extern** »C« erklären, so dürfen Sie das nur bei einer Funktion aus einer Gruppe mit gleichem Namen tun. Sonst würde der Linker mehr als eine globale Funktion mit demselben Namen finden.

Operand of delete must be non-const pointer *Compiler-Fehler*

Der Operand von **delete** muß ein nicht konstanter Zeiger sein

Ein konstanter Zeigerwert darf nicht mit dem Operator **delete** gelöscht werden.

operator [] missing] *Compiler-Fehler*

Im **operator []** fehlt **]**

Der C++ **operator []** wurde als **operator [** deklariert. Ergänzen Sie die fehlende eckige Klammer oder ändern Sie die Deklaration.

operator ->> must return a pointer of a class *Compiler-Fehler*

operator ->> muß einen Zeiger oder eine Klasse liefern

Die C++ Funktion **operator ->>** muß als Ergebnis entweder eine Klasse oder einen Zeiger auf eine Klasse (oder **struct** oder **union**) liefern. In jedem Fall muß sich der Pfeiloperator auf etwas beziehen können.

Operator delete must return void *Compiler-Fehler*

Operator **delete** muß **void** zurückliefern

Dieser überladene C++ Operator **delete** war irgendwie anders deklariert

Operator delete [] must return void *Compiler-Fehler*

Operator **delete []** muß **void** zurückliefern

Der überladene C++ Operator **delete** muß **void** zurückliefern und war hier anders deklariert. Korrekt deklarieren Sie den Operator **delete** so:

- mit einem einzelnen **void***-Parameter oder
- einem zweiten Parameter des Typs size_t

Die zweite Version wird der ersten vorgezogen. Der globale Operator **delete** kann nur mit einem einzelnen Parameter deklariert werden.

Operator must be declared as function *Compiler-Fehler*

Operator muß als Funktion deklariert sein

Ein überladener Operator wurde nicht als Funktion deklariert.

Operator new must have an initial parameter of type size_t *Compiler-Fehler*

Operator **new** muß einen Parameter vom Typ size_t haben

Der Operator **new** kann mit einer beliebigen Anzahl von Parametern deklariert werden. Für die Speicherplatzreservierung muß immer mindestens ein Argument vorhanden sein.

Operator new must return an object of type void * *Compiler-Fehler*

Der **new** Operator muß ein Objekt vom Typ **void*** zurückliefern

Dieser überladene Operator **new** war hier anders deklariert.

Operator new [] must have an initial parameter of type size_t *Compiler-Fehler*

Operator **new []**muß einen Parameter vom Typ size_t haben

Der Operator **new** kann mit einer beliebigen Anzahl von Parametern deklariert werden. Für die Speicherplatzreservierung muß immer mindestens ein Argument vorhanden sein.

Operator new [] must return an object of type void * *Compiler-Fehler*

Der **new []** Operator muß ein Objekt vom Typ **void*** zurückliefern

Dieser überladene Operator **new** war hier anders deklariert.

Operators may not have default argument values *Compiler-Fehler*

Operatoren dürfen keine voreingestellten Argumentwerte haben

Überladene Operatoren dürfen keine voreingestellten Argumentwerte haben.

Out of memory *Compiler-Fehler*

Nicht genügend Arbeitsspeicher

Der gesamte Arbeitsspeicher wurde verbraucht. Die Datei sollte auf einem Rechner mit mehr Arbeitsspeicher compiliert werden.

Out of memory *Bibliotheken-Fehler*

Nicht genügend Arbeitsspeicher

Aus irgendwelchen Gründen hatten TLIB oder die C++ IDE nicht mehr genügend Speicher während der Erzeugung der Bibliothek zur Verfügung. Bei vielen besonderen Ursachen wird eine detailliertere Meldung ausgegeben.

Schließen Sie eine oder mehrere Anwendungen, oder versuchen Sie, zusätzlichen Speicher durch Schließen einiger Texteditoren zu erlangen

Out of memory *Linker-Fehler*

Nicht genügend Arbeitsspeicher

TLINK hat nicht genügend Arbeitsspeicher zur Verfügung. Dies bedeutet normalerweise daß, zuviele Module, externals, Gruppen oder Segmente von den Objektdateien, die verkettet werden, definiert sind. Versuchen Sie den Speicher für Festplattencaches und/oder Ram-Laufwerken zu verkleinern.

Out of memory creating extended dictionary *Bibliotheken-Fehler*

Ungenügend Arbeitsspeicher beim Erstellen eines erweiterten Verzeichnisses

Die Bibliothek ist erstellt, aber wird kein erweitertes Verzeichnis enthalten

Out of memory for block *Block* *Linker-Fehler*

Zuwenig Arbeitspeicher für Block *Block*

TLINK hat nicht genügend Arbeitsspeicher zur Verfügung. Der Fehler sollte eigentlich nicht auftreten, bitte melden Sie dies Borland.

Out of memory reading LE/LIDATA record from object module *Bibliotheken-Fehler*

Nicht genügend Speicher, LE/LIDATA-Record aus Objekt-Modul kann nicht gelesen werden

Die Bibliotheksverwaltung versucht einen Daten-Record aus dem Objekt-Modul einzulesen, verfügt jedoch nicht über den notwendigen Speicher. Falls das hinzuzufügende Modul eines oder mehrere große Datensegmente besitzt, sollten Sie versuchen, das Modul vor allen anderen Modulen hinzuzufügen.

Out of space allocating per module debug struct *Bibliotheken-Fehler*
Speicher für Modul-Debugging-Struktur kann nicht reserviert werden
Die Bibliotheksverwaltung konnte den Speicher für die Debugging-Informationen eines bestimmten Objekt-Moduls nicht belegen. Das Problem kann gelöst werden, indem die Debugging-Informationen einiger Module, die der Bibliothek hinzugefügt werden sollen, entfernt werden.

Output device is full *Bibliotheken-Fehler*
Kein Platz auf Ausgabe-Gerät
Dieser Fehler deutet normalerweise darauf hin, daß die Platte/Diskette voll ist.

Overlays generated and no overlay manager included *Linker-Warnung*
Overlays generiert, aber kein Overlay-Manager eingefügt
Diese Warnung erscheint, wenn Overlays angefertigt werden, aber das Symbol _OVRTRAP_ in keiner der Objekt-Module oder Bibliotheken, in die es verkettet wird, definiert wird.

Overlays only supported in medium, large and huge memory models *Compiler-Fehler*
Overlays nur unterstützt in medium, large und **huge** Speichermodellen
Es werden nur Programme overlaid die medium, large und **huge** Speichermodelle benutzen.

Overload is now unnecessary and obsolete *Compiler-Warnung*
Das Schlüsselwort **overload** ist jetzt überflüssig
Frühere Versionen von C++ brauchten den Begriff **overload** für überladene Funktionsnamen. C++ verwendet jetzt »typensichere Bindung«, bei der alle Funktionen als überladen angesehen werden, falls sie nicht anders markiert sind. **overload** sollte nicht mehr verwendet werden.

Overloadable operator expected *Compiler-Fehler*
Überladbarer Operator erwartet
Fast alle C++ Operatoren können überladen werden. Die einzigen nicht überladbaren sind der Punkt für die Feldauswahl (**.**), der Punkt-Stern (**.***), der doppelte Doppelpunkt (**::**) und der bedingte Ausdruck (**?:**). Die Operatoren # und ## des #Präprozessors sind keine Operatoren der Sprache C oder C++ und können deshalb nicht überladen werden. Interpunktionszeichen wie der Strichpunkt (**;**) sind keine Operatoren und können natürlich auch nicht überladen werden.

Overloaded *Funktionsname* **ambiguous in this context** *Compiler-Fehler*
Überladener *Funktionsname* ist in diesem Kontext mehrdeutig
Ein überladener Funktionsname kann nur dann ohne tatsächlichen Aufruf der Funktion verwendert werden, wenn eine Variable oder ein Parameter eines entsprechenden Typs damit initialisiert wird. Im vorliegenden Fall wurde ein überladener Funktionsname in einem anderen Kontext verwendet.

Overloaded prefix operator *Operator* **used as a postfix operator** *Compiler-Warnung*
Der überladene Präfix-Operator *Operator* wurde als Suffix-Operator verwendet
In der aktuellen C++ Version kann sowohl die Präfix- als auch die Suffix-Version der Operatoren **++** und **--** überladen werden. Wenn älterer Programmtext, in dem nur der Präfix-Operator überladen ist, in einr Suffix-Umgebung eingesetzt wird, verwendet Borland C++ den Präfix-Operator und gibt diese Warnung aus.

P1001 Unable to read file *Dateiname* *Help-Projektdatei-Fehler*
Datei *Dateiname* kann nicht gelesen werden
Die in der Projektdatei angegebene Datei kann nicht gelesen werden: DOS-Fehler.

P1003 Invalid path specified in Root option *Help-Projektdatei-Fehler*
Ungültige Pfadangabe in der Root-Option
Der in der Root-Option angegebene Pfad kann nicht gefunden werden. Der Compiler verwendet das aktuelle Arbeitsverzeichnis.

P1005 Path and filename exceed limit of 79 characters *Help-Projektdatei-Fehler*
Pfad- und Dateiname länger als 79 Zeichen

Der absolute Pfadname bzw. die Kombination aus dem Wurzelverzeichnis und relativem Pfadnamen überschreitet die von DOS festgelegte Maximallänge von 79 Zeichen. Die Datei wurde übersprungen.

P1007 Root path exceeds maximum limit of 66 characters *Help-Projektdatei-Fehler*

Root-Pfadname länger als 66 Zeichen

Der Pfad zum angegebenen Arbeitsverzeichnis überschreitet die von DOS festgelegte Maximallänge von 66 Zeichen. Der Pfadname wird ignoriert und der Compiler verwendet das aktuelle Arbeitsverzeichnis.

P1009 [*Dateien*] section missing *Help-Projektdatei-Fehler*

Der [*Dateien*]-Abschnitt fehlt

Der [*Dateien*]-Abschnitt ist obligatorisch. Die Compilierung wird abgebrochen.

P1011 Option *Optionsname* previously defined *Help-Projektdatei-Fehler*

Option *Optionsname* bereits definiert

Die angegebene Option wurde bereits definiert. Der Compiler ignoriert die neuerliche Definition.

P1013 Project file extension cannot be .HLP *Help-Projektdatei-Fehler*

Die Namenserweiterung der Projektdatei ist unzulässig

Sie können dem Compiler keine Projektdatei mit der Namenserweiterung HLP angeben. Projektdateien haben normalerweise die Dateinamenserweiterung HPJ.

P1015 Unexpected end-of-file *Help-Projektdatei-Fehler*

Unerwartetes Dateiende

Der Compiler ist unerwarteterweise an das Ende der Projektdatei gelangt. Vermutlich fehlt in einer einzubindenden Datei oder in der Projektdatei selbst ein abschließendes Kommentarzeichen.

P1017 Parameter exceeds maximum length of 128 characters *Help-Projektdatei-Fehler*

Parameter länger als 128 Zeichen

Eine Option, ein Kontextstring oder eine Kontextnummer, ein symbolischer Name oder ein anderer Parameter in der angegebenen Zeile ist länger als 128 Zeichen. Die Zeile wird ignoriert.

P1021 Context number already used in [*MAP*] section *Help-Projektdatei-Fehler*

Kontextnummer bereits im [*Map*]-Abschnitt verwendet

Die Kontextnummer in der angegebenen Zeile der Projektdatei wurde vorher bereits einem anderen Kontextstring zugeordnet. Die Zeile wird ignoriert.

P1023 Include statements nested too deeply *Help-Projektdatei-Fehler*

Include-Anweisungen zu tief geschachtelt

Die **#include**-Anweisung in der angegebenen Zeile hat die maximal zulässige Schachtelungstiefe von fünf Stufen überschritten.

P1025 Section heading *Abschnittname* unrecognized *Help-Projektdatei-Fehler*

Unzulässiger *Abschnittname*

Es wurde ein Abschnittkopf verwendet, der nicht vom Compiler unterstützt wird. Die Zeile wird übergangen.

P1027 Bracket mission from section heading sectionname *Help-Projektdatei-Fehler*

Im Abschnitt Abschnittname fehlt die rechte eckige Klammer

In dem angegebenen Abschnittkopf fehlt die rechte eckige Klammer (**]**). Fügen Sie diese ein und compilieren Sie erneut.

P1029 Section heading missing *Help-Projektdatei-Fehler*

Abschnittkopf fehlt

Der Abschnittkopf in der angegebenen Zeile ist nicht vollständig. Dieser Fehler wird auch gemeldet, wenn der erste Eintrag in einer Projektdatei kein Abschnittkopf ist. Der Compiler fährt mit der Bearbeitung in der nächsten Zeile fort.

P1030 Section *Abschnittname* previously defined *Help-Projektdatei-Fehler*

Abschnitt *Abschnittname* bereits definiert

In der Projektdatei ist ein Abschnitt doppelt vorhanden. Die Zeilen unter dem wiederholten Abschnittkopf werden ignoriert und der Compiler setzt seine Bearbeitung beim nächsten gültigen Abschnitt fort.

P1031 Maximum number of build tags exceeded *Help-Projektdatei-Fehler*
Zuviele symbolische Namen vorhanden
Die maximale Anzahl für symbolische Namen beträgt 30. Die überzähligen Namen werden ignoriert.

P1033 Duplicate build tag in [*BUILDTAGS*] section *Help-Projektdatei-Fehler*
Doppelter symbolischer Name im [*BUILDTAGS*]-Abschnitt
Im [*BUILDTAGS*]-Abschnitt ist derselbe symbolische Name doppelt vorhanden.

P1035 Build tag length exceeds maximum *Help-Projektdatei-Fehler*
Symbolischer Name länger als 32 Zeichen
Der symbolische Name in der angegebenen Zeile ist länger als 32 Zeichen. Der Compiler übergeht diesen Eintrag.

P1037 Build Tag *Kennungsname* contains invalid characters *Help-Projektdatei-Fehler*
Der symbolische Name *Kennungsname* enthält unzulässige Zeichen
Symbolische Namen dürfen nur alphanumerische Zeichen und den Unterstrich (_) enthalten. Die Zeile wird übergangen.

P1039 [*BUILDTAGS*] section missing *Help-Projektdatei-Fehler*
[*BUILDTAGS*]-Abschnitt fehlt
Die Option **BUILD** deklarierte eine bedingte Compilierung, aber in der Projektdatei ist kein [*BUILDTAGS*]-Abschnitt vorhanden. Alle Themen werden folglich compiliert.

P1043 Too many tags in Build expression *Help-Projektdatei-Fehler*
Zu viele symbolische Namen im Build-Ausdruck
Der Build-Ausdruck in der angegebenen Zeile enthält mehr als die 20 zulässigen symbolischen Namen. Der Compiler ignoriert die Zeile.

P1045 [*ALIAS*] section found after [*MAP*] section *Help-Projektdatei-Fehler*
[*ALIAS*]-Abschnitt steht nach [*MAP*]-Abschnitt
Der [*ALIAS*]-Abschnitt muß - wenn er verwendet wird - vor dem [*MAP*]-Abschnitt stehen. Andernfalls wird er übergangen.

P1047 Context string *Kontextname* already assigned an alias *Help-Projektdatei-Fehler*
Kontextstring *Kontextname* hat bereits einen Alias-Namen
Folgende Zuweisung ist nicht möglich: a=b und dann a=c. Ein Kontextring kann nur einen Alias-Namen haben. Der angegebene Kontextstring wurde im vorausgehenden [ALIAS]-Abschnitt zugewiesen. Die angeforderte Neuzuweisung in dieser Zeile wird ignoriert.

P1049 Alias string aliasname already assigned *Help-Projektdatei-Fehler*
Alias-String hat bereits einen Alias-Namen
Folgende Zuweisung ist nicht möglich: a=b und dann b=c. Sie können einem Alias-Namen keinen Alias-Namen zuweisen.

P1051 Context string *Kontextname* cannot be used as alias string *Help-Projektdatei-Fehler*
Der Kontextstring *Kontextname* kann nicht als Alias-String verwendet werden
Folgende Zuweisung ist nicht möglich: a=b und dann c=a. Ein Kontextstring, dem ein Aliasname zugeordnet wurde, kann später nicht als Aliasname für einen anderen Kontextstring verwendet werden.

P1053 Maximum number of font ranges exceeded *Help-Projektdatei-Fehler*
Maximale Anzahl der Schriftgrößenbereiche überschritten
Die maximale Anzahl für Schriftgrößenbereiche beträgt 5. Der Rest wird ignoriert.

P1055 Current font range overlaps previously defined range *Help-Projektdatei-Fehler*
Aktueller Schriftgrößenbereich überlappt vorherigen Bereich
Ein Schriftgrößenbereich überlappt eine zuvor durchgeführte Abbildung. Passen Sie die Schriftgrößenbereiche an, um Überlappungen zu entfernen. Die zweite Abbildung wird ignoriert.

P1056 Unrecognized font name in Forcefont option *Help-Projektdatei-Fehler*
Unbekannter Schriftname in der Forcefont-Option
Ein vom Compiler angetroffener Schriftname wird nicht unterstützt. Der Schriftname wird ignoriert und der Compiler verwendet die voreingestellte Helvetica-Schrift.

P1057 Font name too long *Help-Projektdatei-Fehler*
Schriftname zu lang
Der Schriftname darf nicht länger als 20 Zeichen sein. Die Schrift wird ignoriert.

P1059 Invalid multiple-key syntax *Help-Projektdatei-Fehler*
Fehlerhafte Syntax in der ***MULTIKEY***-Option
Die Syntax in der ***MULTIKEY***-Option ist nicht korrekt.

P1061 Character already used *Help-Projektdatei-Fehler*
Zeichen bereits verwendet
Der Bezeichner für die zusätzliche Schlüsselworttabelle ist bereits vergeben. Wählen Sie ein anderes Zeichen.

P1063 Characters K and k cannot be used *Help-Projektdatei-Fehler*
K und **k** dürfen nicht verwendet werden
Diese beiden Zeichen sind für die reguläre Schlüsselworttabelle reserviert. Wählen Sie ein anderes Zeichen.

P1065 Maximum number of keyword tables exceeded *Help-Projektdatei-Fehler*
Zu viele Schlüsselworttabellen vorhanden
Es sind mehr als die zulässigen fünf Schlüsselworttabellen vorhanden. Die überzähligen Tabelle werden ignoriert.

P1067 Equal sign missing *Help-Projektdatei-Fehler*
Gleicheitszeichen fehlt
In einer Option in der angegebenen Zeile fehlt das erforderliche Gleichheitszeichen. Überprüfen Sie die Syntax dieser Option.

P1069 Context string missing *Help-Projektdatei-Fehler*
Kontextstring fehlt
In der angegebenen Zeile fehlt vor einem Gleichheitszeichen ein Kontextstring.

P1071 Incomplete line in *Abschnittname* section *Help-Projektdatei-Fehler*
Unvollständige Zeile im Abschnitt *Abschnittname*
Der Eintrag in der angegebenen Zeile ist nicht vollständig. Die Zeile wird übergangen.

P1073 Unrecognized option in [*OPTIONS*] section *Help-Projektdatei-Fehler*
Unbekannte Option im [*OPTIONS*]-Abschnitt
Es wurde eine vom Compiler nicht unterstützte Option verwendet. Die Zeile wird übergangen.

P1075 Invalid build expression *Help-Projektdatei-Fehler*
Unzulässiger Build-Ausdruck
Die Syntax im Build-Ausdruck in der angegebenen Zeile enthält einen oder mehrere logische oder syntaktische Fehler.

P1077 Warning level must be 1, 2, or 3 *Help-Projektdatei-Fehler*
Warning-Stufe muß 1, 2 oder 3 lauten
Die ***WARNING***-Stufe kann auf 1, 2 oder 3 gesetzt werden. Gemäß Voreinstellung verwendet der Compiler die Stufe 3 (alle Warnungen).

P1079 Invalid compression option *Help-Projektdatei-Fehler*
Fehlerhafte *Compress*-Option
Die Option ***COMPRESS*** kann nur auf ***TRUE*** oder ***FALSE*** gesetzt werden. Der Compiler führt seine Bearbeitung ohne Komprimierung fort.

P1081 Invalid title string *Help-Projektdatei-Fehler*

Ungültiger Titelstring

Die Option ***TITLE*** definiert einen String, der leer ist oder mehr als 32 Zeichen enthält. Im zweiten Fall wird der String entsprechend gekürzt.

P1083 Invalid context identification number *Help-Projektdatei-Fehler*

Ungültige Kontextnummer

Die Kontextnummer in der angegebenen Zeile ist Null oder enthält ungültige Zeichen.

P1085 Unrecognized text *Help-Projektdatei-Fehler*

Unzulässiger Text

Der unzulässige Text in der angegebenen Zeile, der auf korrekten Text folgt, wird ignoriert.

P1086 Invalid font-range syntax *Help-Projektdatei-Fehler*

Fehlerhafte Syntax in der Schriftbereichdefinition

Die Syntax der Schriftbereichdefinition in der angegebenen Zeile ist nicht korrekt. Der Compiler ignoriert die Zeile. Überprüfen Sie die Syntax in der ***MAPFONTSIZE***-Option.

P1089 Unrecognized sort ordering *Help-Projektdatei-Fehler*

Unbekannte Sortierordnung

Sie haben eine vom Compiler nicht unterstützte Sortierordnung angegeben. Wenden Sie sich an die Hotline der Borland GmbH.

Parameter names are used only with a function body *Compiler-Fehler*

Parameternamen brauchen einen Funktionsrumpf

Bei der Deklaration einer Funktion ohne Funktionsrumpf müssen entweder leere Klammern oder ein Funktionsprototyp angegeben werden. Eine Liste nur mit Parameternamen ist nicht erlaubt.

Deklarationsbeispiele:

```
int func();             /* Deklaration ohne Prototyp – ZULÄSSIG */
int func(int, int);     /* Deklaration mit Prototyp – ZULÄSSIG */
int func(int i, int j); /* Parameternamen im Prototyp – ZULÄSSIG */
int func(i, j);         /* Parameternamen allein – NICHT ZULÄSSIG */
```

Parameter ***Nummer*** **missing name** *Compiler-Fehler*

Parametername fehlt

Im Kopf der Funktionsdefinition stand hier nur die Typangabe *Nummer* ohne Parameternamen. In C ist dies unzulässig. (In C++ ist es zwar erlaubt, es ist aber dann kein Zugriff auf den Parameter der Funktion möglich.)

Parameter ***Parameter*** **is never used** *Compiler-Warnung*

Parameter *Parameter* wird nie verwendet

Ein bei der Deklaration der Funktion angegebener Parameter wird innerhalb der Funktion nicht verwendet. Dies kann ein Fehler sein, wird jedoch oft durch falsche Schreibweise des Parameters verursacht. Diese Warnung kann auch auftreten, wenn der Bezeichner im Rumpf einer Funktion erneut als automatische (lokale) Variable deklariert wird. Der Parameter wird von der automatischen Variable überdeckt und nicht verwendet.

Pfad**-path is too long** *Bibliotheken-Fehler*

Pfad zu lang

Dieser Fehler tritt auf, wenn eine Bibliotheksdatei oder der Suchpfad einer Moduldatei mehr als 64 Zeichen enthält.

Pointer to structure required on left side of ->> or ->>* *Compiler-Fehler*

Links vom Pfeil muß ein Zeiger stehen

Auf der linken Seite des Pfeiles (**->>**) ist in C oder C++ nur die Angabe von Zeigern zugelassen. In C++ ist der **->>*** Operator erlaubt.

Possible reference to undefined extern *xxx::i* **in module** *Modulname* *Linker-Warnung*
Mögliche Referenz auf nicht definierte externe Variable xxxx::i in Modul *Modulname*
In der Anwendung wurde ein statisches Datenelement deklariert, es wurde aber keine Definition angegeben.

Possible unresolved external *Symbol* **referenced from module** *Module* *Linker-Warnung*
Möglicherweise wird ein nicht definiertes externes *Symbol* von Modul *Modulname* angesprochen
Diese Warnung erscheint, wenn static-Datenelemente einer Klasse deklariert aber nicht definiert wurden.

Possible use of *Bezeichner* **before definition** *Compiler-Warnung*
Möglicherweise Verwendung von *Bezeichner* vor seiner Definition
In Ihrer Quelltextdatei wurde die Variable *Bezeichner* in einem Ausdruck verwendet, bevor eine Wertzuweisung stattfand. Der Compiler führt eine einfache Programmüberprüfung durch, um diese Fehlerquelle zu finden. Diese Warnung wird generiert, wenn eine Variable physikalisch vor jeglicher Wertzuweisung auftritt. Selbstverständlich kann im aktuellen Programmfluß eine Wertzuweisung erfolgen, bevor diese vom Programm verwendet wird.

Possibly incorrect assignment *Compiler-Warnung*
Möglicherweise inkorrekte Zuweisung
Diese Warnung erhalten Sie, wenn der Compiler einen Zuweisungsoperator als Hauptoperator eines bedingten Ausdrucks (Teil einer **if**-, **while** oder **do while**-Anweisung) antrifft. Diese Warnung wird häufig von einem typographischen Fehler im Gleichheitsoperator ausgelöst. Um die Warnung zu unterdrücken, schreiben Sie die Zuweisung in runde Klammern und vergleichen den gesamten Ausdruck explizit mit Null. Beispielsweise sollte diese Programmzeile:

```
if (a = b) ...
```

so umgeschrieben werden:

```
if ((a = b) != 0) ...
```

Program entry point may not reside in an overlay *Linker-Fehler*
Der Einsprungpunkt des Programms darf nicht in einem Overlay liegen
Obwohl nahezu jeder Teil einer Anwendung als Overlay definiert werden kann, darf die Startadresse nicht in einem Overlay liegen. Diese Fehlermeldung besagt, daß der Versuch unternommen wurde, das Initialisierungsmodul CO *x*.OBJ als Overlay zu definieren, z.B. indem vor dem Startup-Modul die Option **/o** angegeben wurde.

Public *Symbol* **in modul** *Modul1* **clashes with prior module** *Module2* *Bibliotheken-Fehler*
Öffentliches *Symbol* in *Modul1* überschneidet sich mit dem früheren *Modul2*
Ein öffentliches Symbol darf in einer Bibliotheksdatei nur einmal auftreten. Ein zu der Bibliothek hinzugefügtes Modul enthält ein öffentliches *Symbol*, das bereits in einem Modul der Bibliothek vorhanden ist. Das neue Modul kann nicht hinzugefügt werden. In der Meldung wird der Name für *Modul2* angegeben.

Public sy*mbol* **in modul** *Dateiname* **clashes with prior module** *Bibliotheken-Fehler*
Öffentliches *Symbol* im Modul *Dateiname* überschneidet sich mit früherem Modul
Ein öffentliches *Symbol* darf in einer Bibliotheksdatei nur einmal auftreten. Ein zu der Bibliothek hinzugefügtes Modul enthält ein öffentliches Symbol, das bereits in einem Modul der Bibliothek vorhanden ist. Das neue Modul kann nicht hinzugefügt werden.

Public symbol *Symbol* **defined in both module** *Modul1* **and** *Modul2* *Linker-Warnung*
Öffentliches Symbol *Symbol* ist in Modul *Modul1* und Modul *Modul2* definiert
Diese Warnung erscheint, wenn zwei OBJ-Dateien aus der OBJ-Dateienliste dasselbe öffentliche Symbol definieren. Das erste öffentliche Symbol wird das zweite überschreiben. Die Warnung kann mit dem **-wdup** Schalter an- oder ausgeschaltet werden.

R2001 Unable to open bitmap file *Dateiname* *Hilfe-RTF-Fehler*
Bitmap-Datei *Dateiname* kann nicht geöffnet werden
Die angegebene Bitmap-Datei kann nicht gelesen werden: DOS-Fehler.

R2003 unable to include bitmap file *Dateiname* *Hilfe-RTF-Fehler*
Bitmap-Datei *Dateiname* kann nicht aufgenommen werden

Die angegebene Bitmap-Datei konnte nicht gefunden oder gelesen werden. Es handelt sich entweder um einen DOS-Fehler oder es steht nicht genügend Speicherplatz zur Verfügung (»out of memory«).

R2005 Disk full *Hilfe-RTF-Fehler*
Diskette/Platte voll
Die Ressourcen-Datei konnte nicht auf Platte/Diskette geschrieben werden. Schaffen Sie ausreichend Speicherplatz auf dem Ziellaufwerk.

R2009 Cannot use reserved DOS device name for file *Dateiname* *Hilfe-RTF-Fehler*
Reservierter DOS-Gerätename kann nicht als Dateiname verwendet werden
Auf eine Datei wurde mit COM1, LPT2, PRN etc. Bezug genommen. Benennen Sie die Datei um.

R2013 Output file *Dateiname* **already exists as a directory.** *Hilfe-RTF-Fehler*
Ausgabedatei *Dateiname* existiert bereits als Verzeichnis
In dem Verzeichnis, das in der Projektdatei als Arbeitsverzeichnis angegeben wurde, existiert ein Unterverzeichnis mit demselben Namen wie die gewünschte Ressourcendatei. Löschen Sie das Unterverzeichnis oder benennen Sie es um.

R2015 Output file *Dateiname* **already exists as read-only** *Hilfe-RTF-Fehler*
Ausgabedatei *Dateiname* existiert bereits als Nur-Lesen-Datei
Der angegebenen Dateiname kann von der Ressourcendatei nicht überschrieben werden, weil die Datei das Attribut Nur-Lesen hat. Benennen Sie die Projektdatei um oder ändern Sie das Attribut der Datei.

R2017 Path for file *Dateiname* **exceeds limit of 79 characters** *Hilfe-RTF-Fehler*
Pfad der Datei *Dateiname* überschreitet die Länge von 79 Zeichen
Der absolute Pfadname bzw. die Kombination aus Pfad zum Arbeitsverzeichnis und Pfadname für die angegebene Datei überschreitet die von DOS zugelassene Länge von 79 Zeichen. Die Datei wird ignoriert.

R2019 Cannot open file *Dateiname* *Hilfe-RTF-Fehler*
Datei *Dateiname* kann nicht geöffnet werden
Die angegebene Datei kann nicht gelesen werden: DOS-Fehler.

R2021 Cannot find file *Dateiname* *Hilfe-RTF-Fehler*
Datei *Dateiname* kann nicht gefunden werden
Die angegebene Datei konnte nicht gefunden oder gelesen werden. Es handelt sich entweder um einen DOS-Fehler oder es steht nicht genügend Speicherplatz zur Verfügung (»out of memory«).

R2023 Not enough memory to build Help file *Hilfe-RTF-Fehler*
Nicht genügend Speicherplatz für das Erzeugen der Hilfedatei
Schaffen Sie freien Speicherplatz: Entfernen Sie nicht benötigte Anwendungen, Gerätetreiber und speicherresidente Programme aus dem Arbeitsspeicher.

R2025 File environment error *Hilfe-RTF-Fehler*
Datei-Umgebungsfehler
Der Compiler hat nicht genügend Datei-Handles für die Weiterarbeit zur Verfügung. Erhöhen sie die Werte für FILES= und BUFFERS= in der CONFIG.SYS und booten Sie erneut.

R2027 Build tag *Kennungsname* **not defined in [***BUILDTAGS***] section of project file** *Hilfe-RTF-Fehler*
Der symbolische Name *Kennungsname* ist im [*BUILDTAGS*]-Abschnitt der Projektdatei nicht definiert.
Der angegebene symbolische Kennungsname wurde zwar einem Thema zugewiesen, aber in der Projektdatei nicht deklariert. Der Kennungsname wird für das Thema ignoriert.

R2033 Context string in Map section not defined in any topic *Hilfe-RTF-Fehler*
Kontextstring im *MAP*-Abschnitt in keinem Thema definiert
In der Projektdatei sind ein oder mehrere Kontextstrings definiert, der Compiler kann aber keine entsprechenden Themen finden.

R2035 Build expression missing from project file *Hilfe-RTF-Fehler*
Build-Ausdruck nicht in der Projektdatei vorhanden
Die Themen haben symbolische Namen, aber es gibt keinen Build-Ausdruck in der Projektdatei. Der Compiler schließt alle Themen in die Compilierung ein.

R2037 File *Dateiname* cannot be created, due to previous error(s) *Hilfe-RTF-Fehler*
Datei *Dateiname* kann aufgrund vorausgehender Fehler nicht erstellt werden
Die Ressourcen-Datei kann nicht erstellt werden, weil der Compiler keine Themen gefunden hat, die bearbeitet werden können. Korrigieren Sie die diesem Fehler vorausgehenden Fehler und compilieren Sie danach erneut.

R2039 Unrecognized table formatting in topic *Themennummer* of file *Dateiname* *Hilfe-RTF-Fehler*
Unbekannte Tabellenformatierung in Thema *Themennummer* in der Datei *Dateiname*
Der Compiler ignoriert die Tabellenformatierung, die vom Hilfesystem nicht unterstüzt wird. Formatieren Sie die Einträge als fortlaufenden Text neu, falls dies möglich ist.

R2041 Jump *Kontext_String* unresolved in topic *Themennummer* of file *Dateiname* *Hilfe-RTF-Fehler*
Sprung-*Kontext-String* im Thema *Themennummer* in der Datei *Dateiname* nicht auswertbar
Das angegebene Thema enthält einen Kontext-String, der ein nicht vorhandenes Thema kennzeichnet. Prüfen Sie auf Schreibfehler, damit das gewünschte Thema in die Compilierung eingeschlossen wird.

R2043 Hotspot text cannot spread over paragraphs *Hilfe-RTF-Fehler*
Der Kontaktpunkt darf sich nicht auf mehrere Absätze beziehen
Ein Kontaktpunkt erstreckt sich auf zwei Absätze. Entfernen Sie die Absatzmarkierung oder ändern Sie die Formatierung des Kontaktpunktes.

R2045 Maximum number of tab stops reached in topic *Themennummer* of file *Dateiname* *Hilfe-RTF-Fehler*
Maximale Anzahl von Tabulatoren im Thema *Themennummer* der Datei *Dateiname* erreicht
Im angegebenen Thema sind mehr als die 32 zulässigen Tabulatoren vorhanden. Nach der 32. Tabulatorstelle wird die voreingestellte Tabulierung verwendet.

R2047 File *Dateiname* not created *Hilfe-RTF-Fehler*
Datei *Dateiname* nicht angelegt
Es sind keine Themen für die Compilierung vorhanden oder der Build-Ausdruck ist für alle Themen False. Es wird keine Ressourcen-Datei erstellt.

R2049 Context string text too long in topic *Themennummer* of file *Dateiname* *Hilfe-RTF-Fehler*
Der Kontextstring im Thema *Themennummer* der Datei *Dateiname* ist zu lang
Der verborgene Kontextstring darf nicht länger als 64 Zeichen sein. Der String wird ignoriert.

R2051 File *Dateiname* ist not a valid RTF topic file *Hilfe-RTF-Fehler*
Die Datei *Dateiname* ist keine gültige RTF-Themendatei
Die angegebene Datei ist keine gültige RTF-Datei. Überprüfen Sie, ob Sie die Themen mit Ihrem Textprogramm im RTF-Format abgespeichert haben.

R2053 Font *Schriftname* in file *Dateiname* not in RTF font table *Hilfe-RTF-Fehler*
Die Schrift *Schriftname* in der Datei *Dateiname* befindet sich nicht in der RTF-Schriftentabelle
Das Thema wurde in einer Schrift erfaßt, die nicht im RTF-Kopf definiert ist. Der Compiler verwendet die voreingestellte Schrift.

R2055 File *Dateiname* is not a usable RTF topic file *Hilfe-RTF-Fehler*
Die Datei *Dateiname* ist keine verwendbare RTF-Themendatei
Die angegebene Datei enthält einen gültigen RTF-Kopf, der Inhalt entspricht aber nicht oder nur teilweise dem RTF-Format.

R2057 Unrecognized graphic format in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Unbekanntes Grafikformat im Thema *Themennummer* der Datei *Dateiname*
Der Compiler unterstützt nur Windows-Bitmaps. Stellen Sie sicher, daß keine Meta-Dateien oder Macintosh-Formate verwendet wurden. Die Grafik wird ignoriert.

R2059 Context string identifier already defined in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der Kontextstring-Bezeichner ist bereits im Thema *Themennummer* der Datei *Dateiname* definiert
Es gibt mehr als eine Bezeichner-Fußnote für den Kontextstring des angegebenen Themas. Der Compiler verwendet den in der ersten Fußnote (#) definierten Bezeichner.

R2061 Context string *Kontextname* **already used in file** *Dateiname* *Hilfe-RTF-Fehler*
Der Kontextstring *Kontextname* ist schon in der Datei *Dateiname* vorhanden
Der angegebene Kontextstring wurde zuvor bereits einem Thema zugeordnet. Der Compiler ignoriert den folgenden String, und das Thema hat damit keinen Bezeichner.

R2063 Invalid context-string identifier for topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Unzulässiger Kontextstring-Bezeichner für das Thema *Themennummer* der Datei *Dateiname*
Die Kontextstring-Fußnote enthält Zeichen, die nicht alphanumerisch sind, oder sie ist leer. Dem Thema wird kein Bezeichner zugewiesen.

R2065 Context string defined for index topic is unresolved *Hilfe-RTF-Fehler*
Für das Indexthema definierter Kontextstring kann nicht gefunden werden
Das Indexthema in der Projektdatei kann nicht gefunden werden. Der Compiler verwendet dafür das erste Thema.

R2067 Footnote text too long in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der Text der Fußnote im Thema *Themennummer* der Datei *Dateiname* ist zu lang
Der Text einer Fußnote darf nicht mehr als 1000 Zeichen umfassen. Die Fußnote wird ignoriert.

R2069 Build tag footnote not at beginning of topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Die Fußnote des symbolischen Kennungsnamens steht nicht am Beginn des Themas *Themennummer* der Datei *Dateiname*
Das angegebene Thema enthält eine Fußnote für einen symbolischen Kennungsnamen, die nicht als erstes Zeichen im Thema steht. Dem Thema wird kein symbolischer Kennungsname zugewiesen.

R2071 Footnote text missing in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Im Thema *Themennummer* der Datei *Dateiname* fehlt der Fußnotentext
Das angegebene Thema enthält eine Fußnote ohne Eintrag.

R2073 Keyword string is null in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der Schlüsselwortstring im Thema *Themennummer* der Datei *Dateiname* ist leer
Für das angegebene Thema existiert eine Schlüsselwortfußnote, die aber keine Zeichen enthält.

R2075 Keyword string too long in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Im Thema *Themennummer* der Datei *Dateiname* ist das Schlüsselwort zu lang
Der Text in der Schlüsselwortfußnote des angegebenen Themas umfaßt mehr als die 255 zugelassenen Zeichen. Die überzähligen Zeichen werden ignoriert.

R2077 Keyword(s) defined without title in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Im Thema *Themennummer* der Datei *Dateiname* haben die Schlüsselwörter keinen Titel
Für das angegebene Thema wurde ein Schlüsselwort (bzw. Schlüsselwörter) definiert, aber dem Thema wurde kein Titel zugewiesen. In der Liste der gefundenen Themen wird dieses Thema ohne Titel angezeigt.

R2079 Browse sequence string is null in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der String der Suchsequenz im Thema *Themennummer* der Datei *Dateiname* ist leer
Die Suchsequenzfußnote für das angegebene Thema enthält keine Sequenzzeichen.

R2081 Browse sequence string too long in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der String der Suchsequenz im Thema *Themennummer* der Datei *Dateiname* ist zu lang
Die Suchsequenzfußnote für das angegebene Thema umfaßt mehr als die zulässigen 128 Zeichen. Die Sequenz wird für das Thema ignoriert.

R2083 Missing sequence number in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Im Thema *Themennummer* der Datei *Dateiname* fehlt die Sequenznummer
Die Suchsequenz für das angegebene Thema endet mit einem Doppelpunkt (:). Löschen Sie den Doppelpunkt oder geben Sie eine »kleine« Sequenznummer ein.

R2085 Sequence number already defined in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Für das Thema *Themennummer* der Datei *Dateiname* existiert bereits eine Sequenznummer
Für das angegebene Thema existiert bereits eine Suchsequenzfußnote. Die folgende Sequenz wird ignoriert.

R2087 Build tag too long *Hilfe-RTF-Fehler*
Der Tag-Name ist zu lang
Der Tag-Name für das angegebene Thema umfaßt mehr als die zulässigen 32 Zeichen. Der Name wird für das Thema ignoriert.

R2089 Title string null in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der Titelstring im Thema *Themennummer* der Datei *Dateiname* ist leer
Die Titelfußnote für das angegebene Thema enthält keine Zeichen. Dem Thema wird kein Titel zugewiesen.

R2091 Title too long in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Der Titel im Thema *Themennummer* der Datei *Dateiname* ist zu lang
Der Titel für das angegebene Thema umfaßt mehr als die zulässigen 128 Zeichen. Die überzähligen Zeichen werden ignoriert.

R2093 Title titlename in topic *Themennummer* **of file** *Dateiname* **used previously** *Hilfe-RTF-Fehler*
Der Titel im Thema *Themennummer* der Datei *Dateiname* wurde bereits verwendet
Der angegebene Titel wurde bereits einem anderen Thema zugewiesen.

R2095 Title defined more than once in topic *Themennummer* **of file** *Dateiname* *Hilfe-RTF-Fehler*
Dem Thema *Themennummer* der Datei *Dateiname* wurden mehrere Titel zugewiesen
Für das angegebene Thema ist mehr als eine Titel-Fußnote vorhanden. Der Compiler verwendet den ersten Titelstring.

R2501 Using old key-phrase table *Hilfe-RTF-Fehler*
Veraltete Schlüsselsatztabelle
Eine maximale Komprimierung kann nur erreicht werden, wenn vor der erneuten Compilierung der Hilfethemen die Datei PH gelöscht wird.

R2503 Out of memory during text compression *Hilfe-RTF-Fehler*
Zu wenig Arbeitsspeicher für Textkomprimierung
Der Compiler hat zur Komprimierung nicht genügend Arbeitsspeicher. Die Compilierung der Ressourcendatei wird ohne Komprimierung fortgesetzt. Entfernen Sie nicht benötigte Anwendungen, Gerätetreiber und speicherresidente Programme.

R2505 File environment error during text compression *Hilfe-RTF-Fehler*
Nicht genügend Datei-Handles für die Komprimierung vorhanden
Dem Compiler stehen nicht genügend Datei-Handles für die Komprimierung zur Verfügung. Die Compilierung der Ressourcendatei wird ohne Komprimierung fortgesetzt. Erhöhen Sie die Werte für FILES= und BUFFERS= in Ihrer CONFIG.SYS-Datei und booten Sie Ihr System erneut.

R2507 DOS file error during text compression *Hilfe-RTF-Fehler*
DOS-Dateifehler während der Komprimierung
Der Compiler hat während der Komprimierung Schwierigkeiten beim Zugriff auf eine Datei. Die Compilierung der Ressourcendatei wird ohne Komprimierung fortgesetzt.

R2509 Error during text compression *Hilfe-RTF-Fehler*

Fehler während der Komprimierung

Es ist ein Komprimierungsfehler aufgetreten: R2503, R2505 oder R2507. Die Compilierung der Ressourcendatei wird ohne Komprimierung fortgesetzt.

R2701 Internal error *Hilfe-RTF-Fehler*
R2703 Internal error *Hilfe-RTF-Fehler*
R2705 Internal error *Hilfe-RTF-Fehler*
R2707 Internal error *Hilfe-RTF-Fehler*
R2709 Internal error *Hilfe-RTF-Fehler*

interne Fehler

Wenden Sie sich an die Hotline der Borland GmbH.

Record kind *Nummer* **found, expected theadr or lheadr in module** *Dateiname* *Bibliotheken-Fehler*

Record-Typ *Nummer* gefunden, im Modul *Dateiname* wurde jedoch theadr oder lheadr erwartet

Die Bibliotheksverwaltung konnte den in die Bibliothek einzufügenden Header-Record des Objektmoduls nicht interpretieren und nimmt an, daß es sich um ein ungültiges Modul handelt.

Record length *Länge* **exceeds available buffer in module** *Modulname* *Bibliotheken-Fehler*

Record-Länge *Länge* größer als verfügbarer Puffer im Modul *Modulname*

Dieser Fehler tritt auf, wenn die Länge *Länge* eines Records größer ist als der zum Laden verfügbare Puffer im Modul *Modulname*. Dieser Fehler erscheint, wenn die Bibliotheksverwaltung nicht mehr genügend dynamischen Speicher zu Verfügung hat.

Record type *Typ* **found, expected theadr or lheadr in** *Modul* *Bibliotheken-Fehler*

Record-Typ *Typ* gefunden; in *Modul* wurde THEADER oder LHEADER erwartet

Die Bibliotheksverwaltung hat im Modul *Modul* anstelle von theadr oder lheadr einen nicht erwarteten Typ *Typ* gefunden.

Redefinition of *Makro* **is not identical** *Compiler-Warnung*

Nicht identische Neudefinition von *Makro*

Ihre Quelltextdatei definiert das Makro *Makro* neu mit einem Text, der nicht vollständig mit der ersten Definition des Makros übereinstimmt. Der alte Text wird durch den neuen ersetzt.

Redefinition of target *Dateiname* *MAKE-Fehler*

Neudefinition des Ziels *Dateiname*

Die angegebene Datei erscheint mehr als einmal auf der linken Seite einer expliziten Regel.

Reference initialized with *Typ1*, **needs lvalue of type** *Typ2* *Compiler-Fehler*

Referenz wurde initialisiert mit *Typ1*, benötigt wird jedoch ein L-Wert vom *Typ2*

Eine Referenz-Variable oder ein Parameter, die nicht als Konstante deklariert sind, müssen mit einem L-Wert des entsprechenden Typs initialisiert werden. In diesem Fall war der Initialisierer entweder kein L-Wert, oder sein Typ paßte nicht zu der zu initialisierenden Referenz.

Reference member *Element* **in class without constructors** *Compiler-Fehler*

Element in Klasse ohne Konstruktor

Eine Klasse, die Referenzelemente enthält, muß mindestens einen vom Benutzer definierten Konstruktor aufweisen. Andernfalls gäbe es keine Möglichkeit, solche Elemente zu initialisieren.

Reference Member *Element* **initialized with a non-reference parameter** *Compiler-Fehler*

Referenzelement *Element* mit Nicht-Referenzparameter initialisiert

Es wurde versucht ein Referenzelement zu einem Konstruktorparameter zu finden. Da der Parameter bei Beendigung des Konstruktors aufhört zu existieren, zeigt dann das Referenzelement auf ein undefiniertes Objekt. (Das ist ein sehr häufiger Fehler, der Programmabstürze und unberechenbares Programmverhalten hervorrufen kann.)

Reference member *Element* is not initialized *Compiler-Fehler*

Referenz-Element *Element* ist nicht initialisiert

Referenzen müssen immer initialisiert werden. Ein Klassenelement vom Referenztyp braucht eine Initialisierungsgröße in allen Konstruktoren dieser Klasse. Das bedeutet, daß Sie sich nicht darauf verlassen können, daß der Compiler auch für eine derartige Klasse Konstruktoren generiert, weil er nicht wissen kann, wie er die Referenzen initialisieren soll.

Reference member *Element* needs a temporary for initialization *Compiler-Fehler*

Referenz-Element *Element* braucht temporäre Größe zur Initialisierung

Sie haben einen Initialisierungswert für einen Referenztyp angegeben, der kein L-Wert des referenzierten Typs war. Der Compiler erzeugt dann eine temporäre Größe für die Initialisierung. Weil es keinen offensichtlichen Platz für diese temporäre Größe gibt, ist diese Initialisierung unzulässig.

Reference variable *Variable* must be initialized *Compiler-Fehler*

Referenzvariable Variable muß initialisiert sein

Das C++ Objekt ist als **const** oder als Variablenbezug deklariert, aber nicht initialisiert. Es muß bei der Deklaration initialisiert werden.

Register allocation failure *Compiler-Fehler*

Nicht genügend Compiler-Register zur Auswertung

Es liegt ein Compiler Fehler vor. Irgend ein Ausdruck in der angedeuteten Funktion ist so komplex, daß der Code-Generator keinen Code erzeugen konnte. Versuchen Sie die Funktion zu vereinfachen. Wenn das nicht funktioniert, kontaktieren Sie die Hotline der Borland GmbH.

Relocation item exceeds 1 MB DOS limit *Linker-Fehler*

Relocation überschreitet die 1-MB-DOS-Grenze

Das DOS-Format für ausführbare Dateien unterstützt keine Relokationseinträge, die 1 MB überschreiten. Obwohl DOS selbst niemals eine ausführbare Datei dieser Größe laden kann, können dies DOS-Erweiterungen leisten. Aus diesem Grund unterstützt TLINK die Erstellung von Dateien, die zu groß für das Laden unter DOS sind. Selbst wenn die Datei unter einem DOS-Extender geladen wird, bleibt die Einschränkung für das erste MB der Datei bestehen.

Relocation offset overflow *Linker-Fehler*

Offset-Überlauf bei Relocation

Dieser Fehler tritt nur bei 32-Bit-Objekt-Modulen auf, und zeigt eine Offset-Relokation (Segment-Fixup), die größer als die DOS Grenze von 64 kB ist, an.

Relocation table overflow *Linker-Fehler*

Relokations-Tabellenüberlauf

Dieser Fehler tritt nur bei 32 Bit-Objekt-Modulen auf. Die Datei, die verkettet werden soll enthält mehr Basis-Fixups als die Standard-DOS-Relokationstabelle aufnehmen kann (Basis fixups entstehen meistens bei Aufrufen von FAR-Funktionen).

Resident Name Table is greater than 64 kB *Linker-Warnung*

Speicherresidente Namenstabelle überschreitet 64 kB

Die Exporte sind zu groß für 64 kB (im Einklang mit dem Industrie-Standard für ausführbare Dateien). Der Linker fährt fort, ignoriert aber alle untergeordneten residenten Namen, die während des Link-Vorgangs auftreten.

Restarting compile using assembly *Compiler-Warnung*

Compilerlauf neu gestartet, unter Verwendung vom Assembler

Der Compiler hat eine **ASM**-Anweisung in Verbindung mit einer **-B**-Kommandozeilenoption oder einer **#pragma-inline**-Anweisung erreicht. Die Compilierung wird unter Verwendung der Assemblier-Fähigkeiten neu gestartet.

Results are safe in file *Dateiname* *Bibliotheken-Warnung*

Ergebnisse in Datei *Dateiname* gesichert

TLIB hat die Bibliothek erfolgreich als temporäre Datei angelegt, kann jedoch die Datei nicht in den gewünschten Bibliotheksnamen umbenennen. Die temporäre Datei wird nicht gelöscht (wodurch die Bibliothek beibehalten werden kann).

Rule line too long *MAKE-Fehler*

Regel-Zeile ist zu lang
Eine Regel war länger als 4.096 Zeichen.

Segment Segment exceeds 64KByte *Linker-Fehler*

Segment Se*gment* überschreitet 64 kB
Diese Meldung erscheint, wenn zuviele Daten für ein Code-Segment deklariert sind, und wenn TLINK Segmente mit gleichem Namen, aber aus verschiedenen Dateien, kombiniert.

Segment Segment is in two groups: *Gruppe1* **and** *Gruppe2* *Linker-Warnung*

Segment Se*gment* geteilt in zwei Gruppen: *Gruppe1* und *Gruppe2*
Der Linker hat zwei konkurrierende Ansprüche der beiden Gruppen gefunden. Normalerweise passiert dies nur in Assembler-Programmen. Es bedeutet, daß zwei Module das Segment zwei verschiedenen Gruppen zugewiesen haben.

Self relative fixup overflowed in module *Modul* *Linker-Warnung*

Relatives Fixup erzeugt einen Überlauf in Modul *Modul*
Diese Meldung erscheint, wenn ein relativer Verweis (call) von einem physikalischen Segment zu einem anderen abgeändert wird. Dieses tritt bei der Verwendung von Assembler-Code häufig auf, aber auch bei Benutzung der Compileroption zur Segmentbenennung. Der Verweis von einem Codesegment zum anderen ist sicher. Wenn jedoch der Verweis vom Codesegment in ein Datensegment geht, so liegt eventuell ein Fehler im Assemblercode vor. Diese Warnung können Sie mit dem Schalter **-w** der TLINK-Optionen ein- bzw. auschalten.

Size of Operator is unknown or zero *Compiler-Fehler*

Die Größe von *Bezeichner* ist unbekannt oder Null
Dieser Bezeichner wurde in einem Zusammenhang verwendet, in dem seine Größe gebraucht wurde. Ein **struct**-Tag darf alleine (noch ohne Definition des **struct**), ein **extern**-Array darf ohne Größenangabe deklariert werden. Dann ist es aber nicht zulässig, sich auf eine derartige Konstruktion zu beziehen (z.B. mit **sizeof**) oder einen Zeiger auf diesen Typ zu dereferenzieren. Verändern Sie die Deklaration so, daß die Größe vom *Bezeichner* verfügbar ist.

sizeof may not be applied to a bit field *Compiler-Fehler*

sizeof darf nicht auf ein Bitfeld angewandt werden
sizeof liefert die Größe eines Datenobjekts in Bytes; dies ist für ein Bitfeld nicht möglich.

sizeof may not be applied to a function *Compiler-Fehler*

sizeof darf nicht auf eine Funktion angewandt werden
sizeof darf nur auf Datenobjekte, nicht auf Funktionen angewandt werden. Die Größe eines Zeigers auf eine Funktion können Sie aber mit **sizeof** ermitteln.

Size of the type is unknown or zero *Compiler-Fehler*

Größe des Typs unbekannt oder 0
Dieser Typ wurde in einem Zusammenhang verwendet, in dem seine Größe gebraucht wurde. Ein **struct**-Tag z.B. darf alleine (noch vor der Definition des **struct**) deklariert werden. Dann ist es aber nicht zulässig, sich auf eine derartige Konstruktion zu beziehen (z.B. mit **sizeof**) oder einen Zeiger auf diesen Typ zu dereferenzieren. Verändern Sie Ihre Deklaration so, daß die benötigte Größe verfügbar ist.

Identifier specifies multiple or duplicate access *Compiler-Fehler*

Bezeichner hat uneinheitliche Zugriffsrechte
Eine Basisklasse kann **public** oder **private** deklariert werden, nicht beides zugleich. Der Zugriffsbezeichner darf für die Basisklasse nur einmal erscheinen.

Stack overflow *Laufzeit-Fehler*

Stack Überlauf
Die voreingestellte Größe des Stack von C++ Programmen ist 5120 Bytes. Dies ist für die meisten Programme ausreichend. Programme mit rekursiven Funktionen oder mit Funktionen, die viele lokale Daten speichern, könnten aber den

Stack zum Überlauf bringen. Diese Meldung erhalten Sie nur, wenn Sie die Option zur Überprüfung des Stack eingeschaltet haben. Benutzen Sie in diesem Fall ein größeres Speichermodell, vergrößern Sie den Stack oder reduzieren Sie das Ausmaß, in dem Ihr Programm vom Stack abhängig ist. Versuchen Sie, zu einem größeren Speichermodell zu wechseln, um den größeren Stack unterzubringen.
Um den Speicherverbrauch von lokalen Variablen reduzieren zu können, betrachten Sie das untere Beispiel. Die Variable *buffer* wurde als statische Variable definiert und verbraucht keinen Stack-Speicherplatz, wie es bei der Variable *list* der Fall ist

```
void anyfunction(void) {
   static int buffer[2000];     /* im Datensegment */
   int list[2000];              /* auf dem Stack   */
}
```

Die Deklaration lokaler Variablen als **static** hat zwei Nachteile:

1. Statische lokale Variablen reduzieren den Heap und den Speicher, der für globale Variablen zur Verfügung steht. Normalerweise kann man das in Kauf nehmen.
2. Die Funktion könnte dadurch nicht mehr wiedereintrittsfähig sein. Wenn die Funktion nämlich rekursiv oder asynchron aufgerufen wird und jeder Aufruf davon abhängt, daß er über eine eigene eindeutige Kopie einer bestimmten Variablen verfügt, dann können Sie diese Variable nicht mehr als statisch deklarieren. In diesem Fall würde jeder Aufruf mit exakt demselben Speicherbereich arbeiten und das Programm könnte völlig unerwatete oder, bei Netz- oder Multitasking- Anwendungen, unvorhersehbare Ergebnisse produzieren, wenn die Funktion sich selbst auszuführen versucht (rekursiv) oder von verschiedenen Programmen zur selben Zeit aufgerufen wird (asynchron). Bei den meisten DOS-Programmen ist dies jedoch kein Problem.

Statement missing ; *Compiler-Fehler*

Abschließendes Semikolon (**;**) fehlt
Der Compiler hat eine Ausdrucksanweisung gefunden, nach der kein Strichpunkt steht.

Storage class *Speicherklasse* **is not allowed here** *Compiler-Fehler*

Speicherklasse *Speicherklasse* ist hier nicht erlaubt
Die angegebene Speicherklasse ist hier nicht erlaubt. Vermutlich wurden zwei Speicherklassen angegeben, obwohl nur eine angegeben werden darf.

String type not allowed with this operand *MAKE-Fehler*

String-Typ mit diesem Operanden nicht erlaubt
Sie haben versucht, einen Operand zu benutzen, mit dem keine String-Typen verglichen werden können. Erlaubte Operanden sind: **==**, **!=**, <, **>**, **<=** und **>=**

Structure passed by value *Compiler-Warnung*

Struktur als Wert übergeben
Diese Warnung wird erzeugt, wenn eine Struktur über ihren Wert als Argument an eine Funktion ohne einen Prototypen übergeben wird. Es ist ein häufiger Programmierfehler, einen Adress-Operator (**&**) bei einer Struktur wegzulassen, wenn diese als Argument übergeben wird. Dieser Fehler ist jedoch akzeptabel, da Strukturen über ihren Wert übergeben werden können. Diese Meldung ermöglicht dem Compiler, Sie vor diesem Fehler zu warnen.

Structure required on left side of .or.* *Compiler-Fehler*

Auf der linken Seite von (**.**) oder (**.***) muß eine Struktur stehen
Die linke Seite eines Punkt-Operators (in C++ auch eines Punkt-Sternchen-Operators) muß als Strukturtyp ausgewertet werden - was nicht der Fall war.

Structure size too large *Compiler-Fehler*

Struktur zu groß

Ihre Quelldatei enthält die Deklaration einer Struktur mit mehr als 64 kB.

Style of function is now obsolete *Compiler-Warnung*

Dieser Stil der Funktionsdefinition ist veraltet

In C++ ist dieser Stil einer C-Funktionsdefinition nicht mehr zulässig:

```
int func(p1, p2)
int p1, p2;
{...}
```

Subscripting missing] *Compiler-Fehler*

Abschließende eckige Klammer fehlt

Der Compiler bemerkte einen Indexausdruck, dem eine schließende eckige Klammer fehlt. Dieser Fehler kann durch falsch positionierte Klammern oder einen überzähligen bzw. fehlenden Operator ausgelöst werden

Superfluous & with function *Compiler-Warnung*

Überflüssiger Adreß-Operator bei Funktion

Der Adreß-Operator (**&**) ist bei einem Funktionsnamen nicht notwendig; derartige Operatoren werden verworfen

Suspicious pointer conversion *Compiler-Warnung*

Möglicherweise unkorrekte Zeigerumwandlung

Der Compiler hat eine Zeigerumwandlung gefunden, die den Zeiger auf einen anderen Typen zeigen läßt. Sie sollten eine Umwandlung durchführen, um die Warnung zu unterdrücken, wenn die Umwandlung tatsächlich richtig ist.

switch selection expression must be of integral type *Compiler-Fehler*

In **switch**-Auswahl nur ganzzahlige Werte

Der Auswahlausdruck in den Klammern einer **switch**-Anweisung muß ganzzahlig sein (**char**, **short** , **int**, **long**, **enum**). Vielleicht ist dies mit einer expliziten Typumwandlung zu erreichen.

switch statement missing (*Compiler-Fehler*

In der **switch**-Anweisung fehlt **(**

Nach dem Schlüsselwort **switch** fehlt in der Anweisung eine öffnende Klammer

switch statement missing) *Compiler-Fehler*

In der **switch**-Anweisung fehlt **)**

Nach der Bedingung fehlt in der **switch-**Anweisung eine schließende Klammer

Dateiname(***Z****eilennummer*)**:Syntax error** *Linker-Fehler*

Dateiname(*Zeilennummer*) falsch

Der Linker hat einen Syntaxfehler in der Modul-Definitionsdatei gefunden; untersuchen Sie Dateinamen und Zeilennummer auf Fehler.

Table limit exceeded *Linker-Fehler*

Überlauf einer internen Tabelle

Diese Meldung resultiert aus dem Überlauf einer internen Linkertabelle. Das bedeutet, daß die zu linkenden Programme die Kapazität des Linkers für die Aufnahme von öffentlichen und externen Symbolen oder logischen Segmentdefinitionen erschöpft haben. Jede Instanz eines unterschiedlichen Segmentnamens in einer Objektdatei zählt als ein logisches Segment; wenn zwei Objektdateien dieses Segment definieren, ergibt dies zwei logische Segmente.

Target index of FIXUP is 0 in module *Modul* *Linker-Fehler*

Zielindex of FIXUP is 0 im Modul *Modul*

Dies ist ein Übersetzungsfehler.

Template argument must be a constant expression *Compiler-Fehler*

Template-Element muß konstanter Ausdruck sein

Ein typenloses Template-Klassenargument muß ein konstanter Ausdruck des entsprechenden Typs sein. Dies beinhaltet konstante, ganzzahlige Ausdrücke und Adressen von Objekten oder Funktionen mit externer Bindung oder externen Elementen.

Template class nesting too deep: *Klasse* *Compiler-Fehler*

Template-Klasse zu tief verschachtelt: *Klasse*

Der Compiler schränkt die Tiefe der Verschachtelung von Template-Klassen ein. Diese Einschränkung wird normalerweise nur überschritten, wenn zwischen den Template-Klassen rekursive Abhängigkeiten bestehen. Wird diese Einschränkung überschritten, so gibt der Compiler für jede der verschachtelten Template-Klassen eine Fehlermeldung aus. Dies vereinfacht normalerweise die Auflösung der Rekursion. Auf diese Fehlermeldung folgt immer der sofortige Abbruch: **Out of memory**.

Als Beispiel siehe folgende Template-Klasse:

```
template<class T> class A
{
   friend  class  B<T*>;
} ;
template<classT> class  B
{
   friend  class  A<T>;
} ;
A<int>  x;
```

Dieser Code würde folgende Fehlermeldung nach sich ziehen:

```
Error: Template class nesting too deep: 'B<int * * * * *>'
Error: Template class nesting too deep: 'A<int * * * *>'
Error: Template class nesting too deep: 'B<int * * * *>'
Error: Template class nesting too deep: 'A<int * * *>'
Error: Template class nesting too deep: 'B<int * * *>'
Error: Template class nesting too deep: 'A<int * *>'
Error: Template class nesting too deep: 'B<int * *>'
Error: Template class nesting too deep: 'A<int *>'
Error: Template class nesting too deep: 'B<int *>'
Error: Template class nesting too deep: 'A<int>'
Fatal: Out of memory
```

Template function argument *Argument* **not used in argument types** *Compiler-Fehler*

Das Argument *Argument* der Template-Funktion befindet sich nicht unter den Argumenttypen

Das gegebene Argument wurde nicht in der Argumentliste der Funktion verwendet. Die Argumentliste einer Template-Funktion muß alle formalen Argumente der Template-Funktion benutzen; ansonsten ist es nicht möglich, eine, auf aktuellen Argumenttypen basierende, Instanz einer Template-Funktion zu generieren.

Template functions may only have type-arguments *Compiler-Fehler*

Template-Funktionen dürfen nur type-Argumente haben

Bei der Deklaration einer Template-Funktion wurde ein Argument verwendet, welches nicht mit **type** eingeleitet wurde. Dies ist bei einer Template-Funktion nicht erlaubt, da der Wert bei einem Aufruf nicht angegeben werden kann.

Templates can only be declared at file level *Compiler-Fehler*

Templates können nur auf Dateiebene deklariert werden

Templates können nicht innerhalb von Klassen oder Funktionen deklariert werden. Sie sind nur im globalen Gültigkeitsbereich oder auf Dateiebene erlaubt.

Templates must be classes or functions *Compiler-Fehler*

Templates müssen Klassen oder Funktionen sein

Die Deklaration eines Template muß entweder einen Klassentyp oder eine Funktion angeben.

Temporary used to initialize *Bezeichner* *Compiler-Warnung*
Verwendung einer temporären Größe um *Bezeichner* zu initialisieren

Temporary used for parameter *Nummer* **in call to** *Funktion* *Compiler-Warnung*
Verwendung einer temporären Größe für Parameter *Nummer* bei Aufruf einer *Funktion*

Temporary used for parameter *Parameter* **in call to** *Funktion* *Compiler-Warnung*
Verwendung einer temporären Größe für Parameter *Parameter* bei Aufruf einer *Funktion*

Temporary used for parameter *Nummer* *Compiler-Warnung*
Verwendung einer temporären Größe für Parameter *Nummer*

Temporary used for parameter *Parameter* *Compiler-Warnung*
Verwendung einer temporären Größe für Parameter *Parameter*
In C++ müssen sich Referenz-Variablen oder -Parameter auf ein Objekt des korrekten Typs beziehen. Stimmen die Typen nicht überein, findet eine Zuweisung des aktuellen Wertes an eine temporäre Größe desselben Typs statt. Die Adresse der temporären Größe wird der Referenzvariable oder dem Referenzparameter zugewiesen. Die Warnung zeigt, daß die Referenzvariable oder der Referenzparameter nicht auf das Erwartete, sondern auf eine temporäre Variable verweist, die sonst nicht benutzt wird.

In dem folgenden Beispiel benötigt die Funktion *f* eine Referenz auf einen **int**, *c* ist **char**:

```
f(int&);
char c;
f(c);
```

Anstatt *f* mit einer Adresse von *c* aufzurufen, generiert der Compiler den Code, der folgendem C++ Quelltext entspricht:

```
int X = c,  f(X);
```

Terminated by user *Linker-Fehler*
Bediener hat abgebrochen
Sie waren es, der den LINK abgebrochen hat.

The »...« handler must be last *Compiler-Fehler*
Die »...«-Behandlungsroutine muß die letzte sein
In einer Liste von **catch**-Behandlungsroutinen, muß die »...«-Routine, sofern vorhanden, die letzte sein, d.h. es dürfen keine weiteren folgen.

The combinations »+*« or »*+« are not allowed *Bibliotheken-Fehler*
Die Kombinationen »**+***« oder »***+**« sind nicht erlaubt
Gleichzeitiges Hinzufügen und Extrahieren eines Objektmoduls aus einer Bibliothek ist nicht zulässig. Die wahrscheinlich gewünschte Aktion benötigt ein '+-'.

The constructor *Konstruktor* **is not allowed** *Compiler-Fehler*
Konstruktor *Konstruktor* hier nicht erlaubt
Konstruktoren der Form *X::(X)* sind nicht erlaubt. Die richtige Schreibweise für einen Kopierkonstruktor ist: *X::(const X&)*.

The value for *Bezeichner* **is not within the range of an int** *Compiler-Fehler*
Der Wert von *Bezeichner* ist kein **int**-Wert
Alle Zählwerte müssen als Integer dargestellt werden können. Sie haben versucht, einen Wert außerhalb des Integerbereichs zuzuweisen. Brauchen Sie in C++ eine Konstante mit diesem Wert, verwenden Sie einen **const** integer.

»this« can be used only within a member function *Compiler-Fehler*
this nur in Elementfunktion möglich
In C++ ist **this** ein reserviertes Wort, das nur innerhalb von Funktionen eines Klassenelements verwendet werden kann.

This initialization is only partly bracketed *Compiler-Warnung*
Initialisierung nur teilweise in Klammern

Das Ergebnis der Auswahl bei Optionen/Projekt/Compiler/Sprachumfang gemäß ANSI. Die Initialisierung ist nur teilweise geklammert. Bei der Initialisierung von Strukturen können Klammern verwendet werden, um die Initialisierung eines jeden Elements der Struktur zu markieren. Ist ein Element selbst ein Array oder eine Struktur, können verschachtelte Klammern verwendet werden. Dies stellt sicher, daß der Compiler jeden Wert dem richtigen Element zuordnet. Der Compiler gibt diese Warnung aus, wenn optionale Klammern weggelassen werden.

Too few arguments in template class name *Template* *Compiler-Fehler*
Zu wenig Argumente in der Template-Klasse *Template*
Dem Namen einer Template-Klasse fehlen aktuelle Werte für einige der formalen Parameter.

Too few parameters in call *Compiler-Fehler*
Aufruf mit weniger Parametern als erforderlich
Ein Funktionsaufruf mit Prototyp (via Funktionszeiger) erfolgte mit zu wenig Parametern. Prototypen erfordern die Übergabe aller Parameter.

Too few parameters in call to *Funktion* *Compiler-Fehler*
Aufruf von *Funktion* mit zu wenig Parametern
Eine zuvor mit einem Prototypen deklarierte bzw. bereits definierte Funktion wird hier mit zu wenig Argumenten aufgerufen.

Too many commas on Command-line *Linker-Fehler*
Zu viele Kommas in einer Befehlszeile
Ein unzulässiger Eintrag in einer Befehlszeile wurde gefunden.

Too many decimal points *Compiler-Fehler*
Zu viel Dezimalpunkte
Es ist eine Gleitkommakonstante mit mehr als einem Dezimalpunkt vorhanden.

Too many default cases *Compiler-Fehler*
Zu viele **default**-Anweisungen
In einer **switch**-Anweisung wurde mehr als eine **default-**Anweisung gefunden.

Too many default libraries *Compiler-Fehler*
Zu viele Standard-Bibliotheken
Der Linker kann maximal 128 Standard-Bibliotheken verwalten.

Too many error or warning messages *Compiler-Fehler*
Zu viele Fehler oder Warnungen
Es können maximal 255 Fehler oder Warnungen auftreten, bis der Compiler abbricht.

Too many error or warning messages *Linker-Fehler*
Zu viele Fehler oder Warnungen
Die Anzahl der Nachrichten vom Compiler haben ihre Grenze überschritten.

Too many errors *Linker-Fehler*
Zu viele Fehler
Der Linker hat mehr Fehler gefunden als die Option **-E** erlaubt

Too many exponents *Compiler-Fehler*
Zu viele Exponenten
Es wurde mehr als ein Exponent in einer Gleitkommakonstante gefunden.

Too many initializers *Compiler-Fehler*
Zu viele Initialisierer
Es sind mehr Initialisierer vorhanden, als durch die initialisierte Deklaration zugelassen sind.

Too many LNAMEs *Linker-Fehler*
Zu viele LNAMEs
TLINK hat ein Limit von 256 LNAMEs in einer .OBJ Datei.

Too many rules for target *Ziel* *MAKE-Fehler*
Zu viele Regeln für Ziel *Ziel*

Too many segments *Linker-Fehler*
Zu viele Segmente
TLINK hat ein Limit von 256 (Segmenten) in einer .OBJ Datei.

Too many storage classes in declaration *Compiler-Fehler*
Zu viele Speicherklassen in der Deklaration
Eine Deklaration darf nicht mehr als eine Speicherklasse enthalten.

Too many suffixes in .SUFFIXES list *MAKE-Fehler*
Zu viele Erweiterungen in der .SUFFIXES-Liste
Sie haben die 255 erlaubten Erweiterungen in der Liste überschritten.

Too many types in declaration *Compiler-Fehler*
Jeweils nur ein Grundtyp zulässig
Eine Deklaration kann nur jeweils einen der folgenden Grundtypen enthalten: **char**, **int**, **float**, **double**, **struct**, **union**, **enum**, **typedef**-*Name.*

Too much global data defined in file *Compiler-Fehler*
Mehr als 64 KByte globaler Daten in einem Modul
Der Gesamtumfang der globalen Datendeklarationen überschreitet 64 KByte. Überprüfen Sie die Deklarationen bei allen Arrays, die zu groß sein könnten. Ziehen Sie auch die Reorganisation des Programms oder die Verwendung von **far**-Variablen in Betracht, wenn alle Deklarationen notwendig sind.

Trying to derive a far class from the huge base *Basisklasse* *Compiler-Fehler*
Es wurde versucht, eine Klasse vom Typ **far** von einer *Basisklasse* vom Typ **huge** abzuleiten
Wenn eine Klasse vom Typ **huge** deklariert (oder so voreingestellt) ist, müssen alle von ihr abgeleiteten Klassen ebenfalls vom Typ **huge** sein.

Trying to derive a far class from the near base *Basisklasse* *Compiler-Fehler*
Es wurde versucht, eine Klasse vom Typ **far** von einer Basisklasse vom Typ **near** abzuleiten
Wenn eine Klasse vom Typ **near** deklariert (oder so voreingestellt) ist, müssen alle von ihr abgeleiteten Klassen ebenfalls vom Typ **near** sein.

Trying to derive a huge class from the far base *Basisklasse* *Compiler-Fehler*
Es wurde versucht, eine Klasse vom Typ **huge** von einer Basisklasse vom Typ **far** abzuleiten
Wenn eine Klasse als Typ **far** deklariert (oder so voreingestellt) ist, müssen alle von ihr abgeleiteten Klassen ebenfalls vom Typ **far** sein.

Trying to derive a huge class from the near base *Basisklasse* *Compiler-Fehler*
Es wurde versucht, eine Klasse vom Typ **huge** von einer Basisklasse vom Typ **near** abzuleiten
Wenn eine Klasse als Typ **near** deklariert (oder so voreingestellt) ist, müssen alle von ihr abgeleiteten Klassen ebenfalls vom Typ **near** sein.

Trying to derive a near class from the far base *Basisklasse* *Compiler-Fehler*
Es wurde versucht, eine Klasse vom Typ **near** von einer Basisklasse vom Typ **far** abzuleiten
Wenn eine Klasse als Typ **far** deklariert (oder so voreingestellt) ist, müssen alle von ihr abgeleiteten Klassen ebenfalls vom Typ **far** sein.

Trying to derive a near class from the huge base *Basisklasse* *Compiler-Fehler*

Es wurde versucht, eine Klasse vom Typ **far** von einer Basisklasse vom Typ **huge** abzuleiten
Wenn eine Klasse als Typ **huge** deklariert (oder so voreingestellt) ist, müssen alle von ihr abgeleiteten Klassen ebenfalls vom Typ **huge** sein.

Two consecutive dots *Compiler-Fehler*

Zwei aufeinanderfolgende Punkte
Eine Auslassung enthält drei Punkte (**...**) ein Dezimalpunkt oder ein Elementauswahl-Operator einen Punkt(**.**). Somit sind zwei aufeinanderfolgende Punkte in einem C-Programm nicht erlaubt.

Two operands must evaluate to the same type *Compiler-Fehler*

Zwei Operanden müssen sich zum selben Datentyp auswerten lassen
Im Bedingungsoperator (**?:**) müssen die Typen der Ausdrücke auf beiden Seiten des Doppelpunkts dieselben sein, außer bei den gängigen Umwandlungen. Einige Beispiele für gängige Umwandlungen:
char nach **int, float** nach **double** oder **void*** in einen bestimmten Zeiger. Hier ergeben die beiden Seiten verschiedene Datentypen, die nicht automatisch konvertiert werden. Unter Umständen brauchen Sie bloß eine Seite in den Datentyp der anderen umzuwandeln.

type *Typ* **is not a defined class with virtual functions** *Compiler-Fehler*

Typ *Typ* ist keine definierte Klasse mit virtuellen Funktionen
Es wurde ein dynamic_cast mit einem Zeiger auf eine Klasse angewendet, die entweder undefiniert ist oder keine virtuellen Elementfunktionen besitzt. In C++ erhalten Sie zu dieser Meldung eine weitere, die Ihnen das Problem genauer erklärt. Sie lautet gewöhnlich: Cannot convert *Typ1* to *Typ2*. Die Gründe für die Typenunverträglichkeit können jedoch vielfältiger Natur sein.

Type mismatch in default argument value *Compiler-Fehler*

Typen-Konflikt im voreingestellten Argumentwert

Type mismatch in default value for parameter *Parameter* *Compiler-Fehler*

Typen-Konflikt im voreingestellten Wert für Parameter *Parameter*
Der voreingestellte Parameterwert konnte nicht in den Typ des Parameters konvertiert werden. Die Meldung »Type mismatch in default argument value« wird verwendet, wenn der Parameter nicht benannt wurde. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in parameter *Nummer* *Compiler-Fehler*

Typ-Konflikt in Parameter *Nummer*
Die über einen Funktionszeiger aufgerufene Funktion wurde über einen Prototypen deklariert. Der mit *Nummer* bezeichnete Parameter (wobei von links nach rechts beginnend mit 1 gezählt wird) konnte nicht in den deklarierten Parametertypen konvertiert werden. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in parameter *Nummer* **in call to** *Funktion* *Compiler-Fehler*

Typkonflikt in Parameter *Nummer* beim Aufruf von *Funktion*
Ihre Quelldatei hat die genannte Funktion über einen Prototypen deklariert. Der mit der Nummer bezeichnete Parameter (wobei von links nach rechts beginnend mit 1 gezählt wird) konnte nicht in den deklarierten Parametertypen konvertiert werden. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in parameter *Parameter* *Compiler-Fehler*

Typkonflikt in *Parameter*
Ihre Quelldatei hat die genannte Funktion über einen Funktionszeiger mit einem Prototypen deklariert. Der mit der Nummer bezeichnete Parameter konnte nicht in den deklarierten Parametertypen konvertiert werden. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in parameter *Parameter* **in call to** *Funktion* *Compiler-Fehler*

Typkonflikt in Parameter *Parameter* beim Aufruf von *Funktion*

Ihre Quelldatei hat die genannte Funktion über einen Prototypen deklariert. Der Parameter konnte nicht in den deklarierten Parametertypen konvertiert werden. Siehe auch den Eintrag bei **Type mismatch in parameter *Parameter*** und die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in parameter *Parameter* **in template class name** *Template* — *Compiler-Fehler*

Typ-Konflikt bei Parameter *Parameter* in Template-Klassenname *Template*
Der für den Parameter angegebene Wert des Template-Arguments stimmt nicht vollständig mit dem Typ des formalen Template-Parameters überein. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in parameter *Nummer* **in template class name** *Template* — *Compiler-Fehler*

Typkonflikt bei Parameter *Nummer* in Template-Klassenname *Template*
Der für den Parameter angegebene Wert des Template-Arguments stimmt nicht vollständig mit dem Typ des formalen Template-Parameters überein. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type mismatch in redeclaration of *Bezeichner* — *Compiler-Fehler*

Neudeklaration von *Bezeichner* mit anderem Typ
Ihre Quelldatei redeklarierte mit einem zur Originalversion unterschiedlichen Typen. Dieser Fehler kann auftreten, wenn eine Funktion aufgerufen und nachfolgend deklariert wird, daß sie einen vom Typ integer **verschiedenen** Wert zurückliefert. In diesem Fall müssen Sie die Funktion deklarieren bevor Sie sie das erste Mal aufrufen. Siehe auch die anderen Bemerkungen zu Typenkonflikten.

Type name expected — *Compiler-Fehler*

Typname erwartet
Einer der folgenden Fehler ist aufgetreten:

- Bei der Deklaration einer Variablen auf Dateiebene oder eines **struct**-Feldes wurde weder Typname noch Speicherklasse angegeben.
- Bei der Deklaration eines **typedef** wurde kein Typname angegeben.
- Bei der Deklaration eines Destruktors für eine C++ Klasse war der Destruktorname kein Typname - er muß zu seiner Klasse passen.
- Bei der Vergabe eines Namens in eine C++ Basisdatei-Klasse war der Name nicht der einer Klasse.

Type qualifier *Bezeichner* **must be a struct or class name** — *Compiler-Fehler*

Der Typqualifizierer *Bezeichner* muß ein Struktur- oder Klassenname sein
Der C++ Qualifizierer in der Konstruktion Qualifizierer::Bezeichner ist nicht der Name einer Struktur oder einer Klasse.

Unable to create output file *Dateiname* — *Compiler-Fehler*

Ausgabedatei *Dateiname* kann nicht erzeugt werden
Die Diskette/Platte, auf der der Compiler die angegebene Datei erzeugen wollte, ist entweder voll oder schreibgeschützt. Dieser Fehler tritt auch auf, wenn das Ausgabeverzeichnis nicht existiert. Wenn die Diskette/Platte schreibgeschützt ist, kopieren sie die Quelldateien auf eine andere Diskette/Platte, und compilieren Sie erneut.

Unable to create turboc:$Ln — *Compiler-Fehler*

Temporäre Datei turboc.$Ln kann nicht erzeugt werden
Der Compiler kann die temporäre Datei TURBOC.$LN nicht erzeugen, weil kein Zugriff auf die Diskette möglich ist oder weil die Diskette voll ist.

Unable to execute command: *Befehl* — *MAKE-Fehler*

Der Befehl *Befehl* kann nicht ausgeführt werden
Ein Befehl wurde nicht ausgeführt. Gründe dafür könnten sein:
Die Kommandodatei konnte nicht gefunden werden; der Befehl wurde falsch geschrieben; kein Plattenspeicher im Swap-Verzeichnis frei; Swap-Verzeichnis existiert nicht oder (wenig wahrscheinlich) das Kommando selbst existiert, wurde aber verfälscht.

Unable to execute command *Befehl* *Compiler-Fehler*

Der Befehl *Befehl* kann nicht ausgeführt werden
TLINK oder TASM kann nicht gefunden werden. Eventuell ist die Diskette schadhaft.

Unable to open file *Dateiname* *MAKE-Fehler*

Datei *Dateiname* kann nicht geöffnet werden

Unable to open *Dateiname* *Linker-Fehler*

Dateiname kann nicht geöffnet werden
Dies geschieht, wenn der Dateiname nicht existiert oder falsch geschrieben wurde.

Unable to open *Dateiname* **for output** *Bibliotheken-Fehler*

Fehler beim Öffnen der Datei *Dateiname* zur Ausgabe
Die Bibliotheksverwaltung konnte die angegebene Ausgabedatei nicht öffnen. Das liegt gewöhnlich am fehlenden Platz auf dem Laufwerk der Zielbibliothek oder an der Listen-Datei. Außerdem tritt dieser Fehler auf, wenn die Zieldatei zwar existiert, aber als Nur-Lese-Datei gekennzeichnet ist.

Unable to open include file *Dateiname* *Compiler-Fehler*

Include-Datei *Dateiname* kann nicht geöffnet werden
Der Compiler konnte die angegebene Include-Datei nicht finden. Mögliche Gründe: Die Datei existiert nicht. Eine **#include**-Datei enthält eine **#include**-Direktive für sich selbst. Sie haben den Parameter FILES in Ihrer CONFIG.SYS im Hauptverzeichnis (erhöhen Sie mindestens auf FILES=20) nicht groß genug gewählt. Überprüfen Sie auch, ob die angegebene Datei existiert.

Unable to open include file *Dateiname* *MAKE-Fehler*

Include-Datei kann nicht geöffnet werden
Der Compiler konnte die angegebene Include-Datei nicht finden. Mögliche Gründe: Die Datei existiert nicht. Eine **!include**-Datei enthält eine **!include**-Direktive für sich selbst. Sie haben den Parameter FILES in Ihrer CONFIG.SYS im Hauptverzeichnis (erhöhen Sie mindestens auf FILES=20) nicht groß genug gewählt. Überprüfen Sie auch, ob die angegebene Datei existiert.

Unable to open input file *Dateiname* *Compiler-Fehler*

Eingabedatei *Dateiname* kann nicht geöffnet werden
Die Quelltextdatei kann nicht gefunden werden. Überprüfen Sie die Schreibweise des Namens und ob sich die Datei auf der richtigen Diskette/Platte oder im richtigen Verzeichnis befindet.

Unable to open makefile *MAKE-Fehler*

MAKE-Datei kann nicht geöffnet werden
Das aktuelle Verzeichnis enthält keine Datei mit Namen MAKEFILE, MAKEFILE.MAK oder die Datei mit der Option **-f.**

Unable to redirect input or output *MAKE-Fehler*

Temporäre Dateien nicht gefunden
MAKE war nicht in der Lage, die temporären Dateien zu öffnen, um Input oder Output umzulenken. Im Netzwerk achten Sie bitte darauf, daß Sie Zugriffsrecht auf das aktuelle Verzeichnis haben.

Unable to rename *Dateiname* **to** *Dateiname* *Bibliotheken-Fehler*

Dateiname kann nicht unbenannt werden
Meist liegt dies am fehlenden Speicherplatz auf der Platte.

Undefined label *Bezeichner* *Compiler-Fehler*

Sprungadresse *Bezeichner* nicht definiert
Die benannte Struktur wurde in der Quelldatei benutzt, wahrscheinlich durch einen Zeiger auf eine Struktur, besaß aber keine Definition in der Quelldatei. Meistens wird dies durch einen falsch buchstabierten Strukturnamen oder eine fehlende Deklaration verursacht.

Undefined structure *Bezeichner* *Compiler-Warnung*

Struktur-Bezeichner nicht definiert

Die angegebene Struktur wurde in der Quelltextdatei verwendet, eventuell in einem Zeiger auf eine Struktur. Es gibt aber keine Definition für diese Struktur im Quelltext. Der Strukturname ist vielleicht falsch geschrieben, oder eine Deklaration fehlt.

Undefined structure *Struktur* *Compiler-Fehler*

Struktur *Struktur* nicht definiert

In Ihrer Quelltextdatei wird die angegebene Struktur einige Zeilen vor dem ausgewiesenen Fehler verwendet (eventuell in einem Zeiger auf eine Struktur), ohne vorher definiert worden zu sein. Entweder enthält der Strukturname einen Tippfehler, oder es fehlt die Deklaration.

Undefined symbol *Bezeichner* *Compiler-Fehler*

Symbol *Bezeichner* ist nicht definiert

Der angegebene Bezeichner ist dem Compiler unbekannt. Mögliche Gründe: Aktuelle Bezeichnerdeklaration ist als Kommentar gekennzeichnet; Tippfehler beim Zugriff oder bei der Deklaration; eine fehlerhafte Deklaration.

Undefined symbol *Symbol* **in module** *Modul* *Linker-Fehler*

Undefiniertes Symbol *Symbol* im Modul *Modul*

Das angegebene Symbol ist im angegebenen Modul referenziert, aber nicht in den gelinkten Objektdateien und Bibliotheken definiert. Überprüfen Sie, ob das Symbol fehlerhaft geschrieben ist.

Dieser Linker-Fehler tritt normalerweise für C oder C++ Symbole auf, wenn eine der folgenden Bedingungen zutrifft: In verschiedenen Quelltextdateien stimmen die Deklarationen eines Symbols vom Typ **pascal** und **cdecl** nicht überein; Sie haben einen für Ihr Programm erforderlichen Namen einer .OBJ-Datei nicht angegeben; Sie haben die Emulationsbibliothek nicht eingebunden.

Wenn Sie C++ Code mit C-Modulen linken, haben Sie eventuell vergessen, externe C-Deklarationen in extern "C" {...} umzuschreiben. Ebenso können zwei Symbole in ihrer Groß-/Kleinschreibung nicht übereinstimmen. Vergleichen Sie die Optionen **/C** und **/c**.

Unexpected } *Compiler-Fehler*

Unerwartete geschweifte Klammer

Unerwartete rechte geschweifte Klammer angetroffen. Suchen Sie nach einer fehlenden **{**.

Unexpected char *X* **in command line** *Bibliotheken-Fehler*

Unerwartetes Zeichen *X* in Befehlszeile

Bibliothek hat einen syntaktischen Fehler in der Befehlszeile entdeckt.

Unexpected end of file *MAKE-Fehler*

Unerwartetes Dateiende

Die Make-Datei ist abgeschlossen, obwohl die Inline-Datei nicht geschlossen wurde.

Unexpected end of file in comment started on *Zeilennummer* *MAKE-Fehler*

Unerwartetes Dateiende im Kommentar, der mit Zeile *Zeilennummer* beginnt

Die Quelltextdatei endet mitten in einem Kommentar. Hier fehlt normalerweise der Abschluß des Kommentars mit (*/).

Unexpected end of file in conditional started on *Zeilennummer* *Compiler-Fehler*

if-Block vor Dateiende nicht abgeschlossen. Anfang der Bedingung in Zeile *Zeilennummer*

Das Ende einer Quelltextdatei wurde erreicht, bevor das zu einem **!if** gehörige **!endif** gefunden wurde. Das **!endif** fehlt oder wurde falsch geschrieben.

Union cannot be a base type *Compiler-Fehler*

Union kann keine Basisklasse sein

Ein union kann nicht als Basistyp für einen anderen Klassentyp benutzt werden.

Union cannot have a base type *Compiler-Fehler*

Union kann keine Basisklasse haben

Im allgemeinen gilt, daß eine C++ Klasse vom Typ Union sein kann, solch eine Klasse kann aber nicht von anderen Klassen abgeleitet werden.

Union member *Element* **is of type class with constructor** *Compiler-Fehler*

Element *Element* einer Variante ist vom Typ Klasse mit Konstruktor

Union member *Element* **is of type class with destructor** *Compiler-Fehler*

Element *Element* einer Variante ist vom Typ Klasse mit Destruktor

Union member *Element* **is of type class with operator=** *Compiler-Fehler*

Element *Element* einer Variante ist vom Typ Klasse mit operator**=**

Eine Klasse vom Typ Union kann keine Elemente vom Typ einer Klasse mit anwenderdefinierten Konstruktoren, Destruktoren oder Operator= enthalten.

Unions cannot have virtual member functions *Compiler-Fehler*

Klassen vom Typ Union dürfen keine virtuellen Elementfunktionen enthalten.

Klassen vom Typ Union darf keine virtuellen Funktionen als Elemente enthalten.

Unknown assembler instruction *Compiler-Warnung*

Unbekannter Assembler-Befehl

Der Compiler hat eine Inline-Assembler-Anweisung gefunden, die einen unzulässigen Opcode enthält

Unknown command line switch *X* **ignored** *Bibliotheken-Warnung*

Unbekannter Kommandozeilen-Schalter *X* wurde übergangen

Ein Schrägstrich (/) wurde in der Kommandozeile oder in einer Kommandodatei entdeckt, ohne daß eine der erlaubten Optionen folgte.

Unknown language, must be C or C++ *Compiler-Fehler*

Sprache unbekannt, C oder C++ erforderlich

In der C++ Konstruktion

```
extern »Name« type func( /*...*/ );
```

muß der angegebene Name der Sprache »C« oder »C++« entstammen (verwenden Sie die Anführungszeichen); andere Sprachen werden nicht akzeptiert. Eine externe Pascalfunktion kann z.B. folgendermaßen deklariert werden, ohne daß der Compiler sie umbenennt:

```
extern »C« int pascal func( /*...*/ );
```

Eine (möglicherweise überladene) C++ Funktion können Sie folgendermaßen als Pascal deklarieren und zugleich das Umbenennen durch den Compiler zulassen (um das Überladen zu erlauben):

```
extern int pascal func( /*...*/ );
```

Unknown option *Linker-Fehler*

Unbekannte Option

In der Befehlszeile oder in einer Kommando-Datei wurde ein (/) , (**-**) oder DOS-switch-Zeichen entdeckt, ohne eine folgende zulässige Option.

Unknown preprocessor directive: *Bezeichner* *Compiler-Fehler*

Unbekannte Präprozessor-Direktive: *Bezeichner*

Der Quelltext enthält eine Zeile, die mit einem **#** beginnt, darauf folgt aber keiner der legalen Direktiven-Namen oder der Rest der Direktive war nicht exakt angegeben.

Unknown preprocessor statement *MAKE-Fehler*

Unbekannter Präprozessor-Bezeichner

Ein **!** Zeichen wurde gefunden am Anfang einer Zeile, dananch steht aber kein **error**, **undef**, **if**, **elif**, **include**, **else**, oder **endif**.

Unreachable code *Compiler-Warnung*

Programmteil kann nie ausgeführt werden

Nach einer **break**-, **continue**-, **goto**- oder **return**- Anweisung folgt kein Label oder kein Ende einer Schleife oder Funktion. Der Compiler überprüft **while**-, **do**- und **for**-Schleifen mit einer konstanten Testbedingung und versucht Schleifen festzustellen, die nie durchlaufen werden. Mit der Kommandozeilenoption **-wrch** können Sie die Meldung kontrollieren. Standardmäßig ist sie ausgeschaltet.

Unsupported option string *Linker-Fehler*

Unzulässiges Options-Zeichen

Sie haben dem Linker eine unzulässige Option deklariert.

Unterminated string or character constant *Compiler-Warnung*

Zeichen/Stringkonstante nicht korrekt abgeschlossen

Nach einer String- oder Zeichenkonstante fehlt das abschließende Anführungszeichen

Use '> >' for nested templates instead of '>>' *Compiler-Fehler*

Benutzen Sie '**> >**' für verschachtelte Templates statt '**>>**'

Ein Whitespace-Zeichen (Leerzeichen, Tabulator, ...) wird benötigt, um das abschließende »**>**«-Zeichen in einer verschachtelten Schablone abzugrenzen. Aber da es ein extrem häufiger Fehler ist, akzeptiert der Compiler auch »**>>**« und erzeugt diese Warnung.

Use . or -> to call *Funktion* *Compiler-Fehler*

Aufruf einer Funktion *Funktion* mit **.** oder **->**

Sie haben versucht eine Elementfunktion aufzurufen, ohne ein Objekt anzugeben.

Use . or -> to call *Element***, or & to take its address** *Compiler-Fehler*

Aufruf einer Elementfunktion *Element* mit **.** oder **->**; Abruf ihrer Adresse mit **&**

Eine Referenz auf ein nicht-statisches Klassenelement ohne Objekt trat auf. Ein derartiges Element kann nur mit Objekt verwendet werden, oder seine Adresse muß durch den Operator **&** übernommen werden.

Use :: to take the address of a member function *Compiler-Fehler*

Adresse einer Elementfunktion nur über den Operator **::**

Ist *f* eine Elementfunktion der Klasse *c*, erhält man deren Adresse über:

```
&c::f
```

Beachten Sie, daß ein Klassenname und nicht der Name eines Objektes verwendet wird, und daß ein doppelter Doppelpunkt zwischen Klasse und Funktion stehen muß. (Zeiger auf Elementfunktionen sind keine echten Zeiger und beziehen sich auf keine besondere Instanz einer Klasse).

Use /e with TLINK to obtain debug information from library *Bibliotheken-Warnung*

Benutzen Sie **/e** um Debug-Information von der Bibliothek zu erhalten

Die Bibliothek wurde mit einem erweiterten Dictionary aufgebaut, und enthält Debugging-Informationen. Um Debugging-Information zu erhalten, müssen Sie mit dem **/e** Schalter den erweiterte Dictionary im Linker ausschalten

Use of : and :: dependents for target *Ziel* *MAKE-Fehler*

Falsche Benutzung von **:** und **::**

Sie haben versucht, das *Ziel* in einfachen und mehrfachen Beschreibungs-Blocks zu benutzen, Beispiele:

```
filea: fileb
filea:: filec
```

Use qualified name to access nested type *Typ* *Compiler-Warnung*

Für den Zugriff auf den verschachtelten Typ *Typ* muß ein qualifizierter Name verwendet werden

In älteren Versionen der C++ Spezifikation waren innerhalb von Klassen deklarierte **typedef**- und Tag-Namen im globalen Gültigkeitsbereich direkt sichtbar. Aufgrund der neuesten C++ Spezifikationen muß diesen Namen **class**::Bezeichner vor-

angestellt werden, wenn sie außerhalb des Gültigkeitsbereichs ihrer Klasse verwendet werden sollen. Der Compiler gibt diese Warnung aus, wenn ein Name eindeutig in einer einzigen Klasse definiert ist. Der Compiler läßt die Verwendung ohne Klasse:: zu. Dies ermöglicht die Compilierung älterer Programme.

User break *Compiler-Fehler*

Abbruch der Compilierung durch Eingabe von [*Strg*]+[*Untbr*]

Sie haben die Compilierung durch Eingabe von [*Strg*]+[*Untbr*] abgebrochen (Dies ist nur eine Bestätigung, keine Fehlermeldung).

Using based linking for DLLs may cause the DLL to malfunction *Linker-Warnung*

Aufruf des Linkers mit dem Schalter **/B** kann bei DLLs zu einer Fehlfunktion führen

Diese Warnung tritt auf, wenn Sie den **/B**-Schalter beim Linken einer DLL setzen. In fast allen Fällen ensteht ein Fehler, und die Anwendung ist nicht lauffähig. Sie können die Warnung mit dem Schalter **-w** kontrollieren.

Value of type void is not allowed *Compiler-Fehler*

Wert vom Typ **void** ist unzulässig

Ein Wert vom Typ **void** ist eigentlich kein Wert und darf deshalb nicht in Fällen verwendet werden, wo ein Wert erforderlich ist. Dazu gehören die rechte Seite einer Zuweisung, des Argumentes einer Funktion und der steuernden Bedingung einer **if**-, **for**- oder **while**-Anweisung.

VIRDEF Name Conflict for *Funktion* *Linker-Fehler*

VIRDEF Funktionsnamen-Konflikt

Der Compiler muß Namen auf eine bestimmte Länge, die für den Linker einzuhalten ist, abschneiden. Daraus können natürlich zwei identische Namen resultieren und wenn diese auch noch **VIRDEF**-Namen sind, dann erscheint diese Meldung. Versuchen Sie die Namen bestmöglichst zu ändern.

Variable *Bezeichner* **is initialized more than once** *Compiler-Fehler*

Variable *Bezeichner* mehrmals initialisiert

Diese Variable wird hier zum wiederholten Mal initialisiert. Auf Dateiebene darf eine Variable zwar mehrmals deklariert, aber nur einmal initialisiert werden (selbst wenn die Initialisierungen identisch sind).

'Virtual' can only be used with member functions *Compiler-Fehler*

virtual nur bei Elementfunktion erlaubt

Ein Datenelement wurde als **virtual** deklariert. Nur Elementfunktionen können als **virtual** deklariert werden.

Virtual function *Funktion1* **conflicts with base class** *Basisklasse* *Compiler-Fehler*

Virtuelle *Funktion1* kollidiert mit Basisklasse *Basisklasse*

Eine virtuelle Funktion hat dieselben Argumenttypen wie die Basisklasse, liefert aber einen anderen Ergebnistyp. Dies ist unzulässig.

Virtual specified more than once *Compiler-Fehler*

Der Begriff **virtual** tritt mehr als einmal auf

In C++ darf das reservierte Wort **virtual** in der Deklaration einer Elementfunktion nur einmal auftreten.

Void& is not a valid type *Compiler-Fehler*

void& ist kein gültiger Datentyp

Eine Referenz bezieht sich immer auf ein Objekt. Ein Objekt kann aber nicht vom Typ **void** sein. Darum ist der Typ **void** hier nicht erlaubt.

Void functions may not return a value *Compiler-Warnung*

void-Funktion darf keinen Wert zurückliefern

Ihre Quelltextdatei deklariert den Rückgabewert der aktuellen Funktion als **void**, der Compiler fand jedoch eine **return**-Anweisung mit einem Wert. Der Wert der **return**-Anweisung wird ignoriert.

Funktion **was previously declared with the language** *Sprache* — *Compiler-Fehler*

Funktion war ursprünglich in *Sprache* deklariert

Einer Funktion kann nur ein Sprachmodifizierer (**cdecl** oder **pascal**) zugeordnet werden. Der Funktion wurden an zwei Stellen verschiedene Modifizierer zugeordnet.

While statement missing (— *Compiler-Fehler*

In der **while** Schleife fehlt **(**

Der Compiler findet in der **while**-Anweisung nach dem Schlüsselwort while keine linke Klammer

While statement missing) — *Compiler-Fehler*

In der **while** Schleife fehlt **)**

Der Compiler findet in der **while-**Anweisung nach der Bedingung keine rechte Klammer. Dieser Fehler tritt auf, wenn TLINK nicht in alle benötigten Dateien schreiben konnte. Die Fehlersuche ist wahrscheinlich eine volle Diskette/Platte.

Write error on file *Dateiname* — *MAKE-Fehler*

Schreibfehler beim Zugriff auf die Datei *Dateiname*

MAKE konnte die in der Make-Datei angegebene DAtei nicht öffenen oder in diese schreiben. Prüfen Sie, ob noch genug Speicher auf der Platte vorhanden ist und ob Sie Schreibzugriff auf die Platte haben.

Wrong number of arguments in call of macro *Makro* — *Compiler-Fehler*

Falsche Anzahl von Argumenten in Aufruf von Makro *Makro*

Der Aufruf des benannten Makros erfolgte nicht mit der korrekten Anzahl von Argumenten.

A N H A N G **B**

Borlands benutzerdefinierte Dialogelemente für Windows

Bevor Sie diesen Anhang lesen, sollten Sie sich mit den Steuerungselementen für Dialogfenster und demDialog-Editor vertraut gemacht haben. Borlands benutzerdefinierte Elemente für Windows (BWCC: Borland Windows Custom Conrols) sind in einer Bibliothek zusammengestellt. Diese enthält eine von Borland vordefinierte Dialogklasse und einige vordefinierte Dialogelemente (z.B. Aktionsschalter oder Schaltfelder). Durch BWCC läßt sich die Oberfläche besser gestalten und die Funktionalität der Dialogfenster wird optimiert.

Zwei der Dateien zum Resource Workshop enthalten zusätzliche Informationen über BWCC:

- BWCCAPI.RW enthält technische Informationen über die Schnittstelle zu Anwendungsprogrammen.
- BWCCSTYL.RW enthält Vorschläge zur Gestaltung eigener Dialogfenster.

Borlands vordefinierte Dialogklasse

Die benutzerdefinierte Dialogklasse BORDLG bietet sowohl visuelle als auch funktionale Erweiterungen:

- Der Hintergrund des Dialogfensters wird mit einem für die gewählte Bildschirmauflösung geeigneten Muster gefüllt. Bei VGA-Bildschirmen und Bildschirmen mit höherer Auflösung wirkt der Hintergrund gesprenkelt; bei EGA- und monochromen Bildschirmen ist der Hintergrund weiß.

- Die Darstellung der Dialogfenster wird optimiert, da die benutzerdefinierten Dialogelemente direkt in das Dialogfenster gezeichnet werden. Dadurch werden die langsameren Windows-Routinen umgangen.

Um BWCC in einem Programm einzusetzen, gehen Sie folgendermaßen vor:

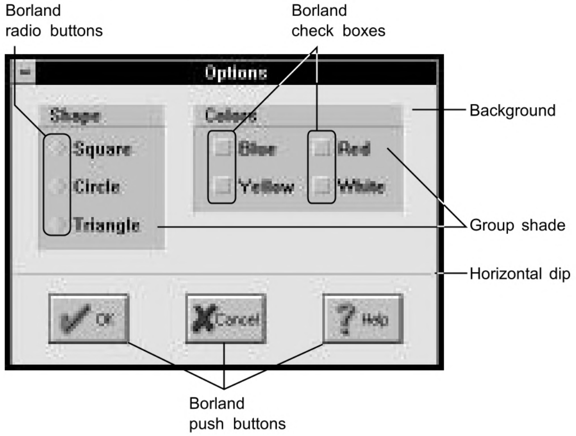
1. Öffnen Sie das Dialogfenster, das Sie konvertieren möchten.
2. Doppelklicken Sie auf den Titel des Dialogfensters, um das Dialogfenster Fenstereigenschaften aufzurufen.
3. Geben Sie im Textfeld Klasse die Bezeichnung b*ordlg* ein, und klicken Sie auf *OK*.

Borlands vordefinierte Dialogelemente

Vordefinierte Dialogelemente verleihen den Dialogfenstern einen 3-D-Effekt und erhöhen ihre visuelle Wirkung. Bezüglich der Funktionalität ergibt sich für den Anwender jedoch kein Unterschied gegenüber den standardmäßigen Windows-Dialogelementen. Die Dialogelemente enthalten aber Erweiterungen, auf die später noch eingegangen wird.

Die folgende Abbildung zeigt ein mit Hilfe von BWCC konvertiertes Dialogfenster. Es enthält mehrere vordefinierte Dialogelemente.

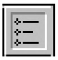
Abb. B.1: Dialogfenster mit Borlands vordefinierten Dialogelementen



Nachfolgend werden die vordefinierten Dialogelemente und die dazugehörigen Symbole aufgelistet. Sie können wie bei standardmäßigen Windows-Dialogelementen die Dialogelemente einfügen, indem Sie sie aus derWerkzeugpalette im Dialog-Editor auswählen.

Die Klasse jedes Dialogelements ist in der Beschreibung angegeben. Um sich die Klasse und andere Eigenschaften des Dialogelements anzeigen

zu lassen, halten Sie die [Strg]-Taste gedrückt und doppelklicken auf das Symbol. Sie erhalten dann das Dialogfenster mit denallgemeinen Elementeigenschaften.



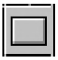
Gruppenfeld: Ein schattiertes Rechteck, das andere Dialogelemente visuell gruppiert. Es kann vertieft oder erhöht dargestellt werden. Es gehört der Klasse BorShade an.



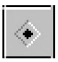
Horizontale Tiefe: In die Oberfläche des Dialogfensters scheint eine Trennlinie eingraviert worden zu sein. Sie können die horizontale Tiefe in eine horizontale Erhöhung umwandeln. Sie gehört der Klasse BorShade an.



Vertikale Tiefe: Siehe horizontale Tiefe. Sie gehört der Klasse BorShade an.



Aktionsschalter: Eine Reihe von auffälligen Aktionsschaltern, die auch durch das übergeordnete Fenster gezeichnet werden können. Die Aktionsschalter von Borland sind größer als die standardmäßigen Windows-Aktionsschalter. Sie gehören der Klasse BorBtn an.



Schaltfelder: Ein erhöhtes rautenförmiges Schaltfeld. Wenn das Schaltfeld angeklickt wird, erscheint eine schwarze Raute in seiner Mitte, und die Schattierung des Schaltfelds wird so verändert, daß der Eindruck eines gedrückten Schalters entsteht. Auch dieses Element kann durch das übergeordnete Fenster gezeichnet werden. Es gehört der Klasse BorRadio an.



Markierungsfeld: Ein erhöhtes Markierungsfeld, in dem statt eines Kreuzchens ein Häkchen stehen kann. Dieses Element kann auch durch das übergeordnete Fenster gezeichnet werden. Es gehört der Klasse BorCheck an.



Statischer Text: Ein festzulegender Text, der hauptsächlich für die Beschreibung von Teilen des Dialogfenster verwendet wird. Dieses Element gehört der Klasse BorStatic.

Erweiterungen zu Schaltern und Markierungsfeldern

Die von Borland vordefinierten Aktionsschalter, Schaltfelder und Markierungsfelder sind gegenüber den standardmäßigen Windows-Elementen in den folgenden Punkten erweitert worden:

- Das übergeordnete Fenster hat einen erweiterten Zugriff auf den Tastaturfokus und den Tabulator und erhält zusätzliche Meldungen. Falls Sie die Option Ben*achrichtigen* im Dialogfenster Element*eigenschaften* an-

wählen, sendet das entsprechende Element zur Laufzeit des Programms folgende Botschaften:

- BBN_SETFOCUSMOUSE: Diese Botschaft wird von dem Dialogelement an das übergeordnete Fenster gesendet, wenn der Schalter durch Anklicken den Tastaturfokus erhält.
 - BBN_SETFOCUS: Diese Botschaft wird von dem Dialogelement an das übergeordnete Fenster gesendet, wenn der Schalter durch eine andere Aktion als einen Mausklick den Tastaturfokus erhält.
 - BBN_GOTATAB: Diese Botschaft wird von dem Dialogelement an das übergeordnete Fenster gesendet, wenn der Anwender [*Tab*] drückt, während das Dialogelement den Tastaturfokus hat. Das übergeordnete Fenster kann in die Verarbeitung der Eingabe eingreifen, indem ein von Null verschiedener Wert zurückgegeben wird.
 - BBN_GOTABTAB: Diese Botschaft wird von dem Dialogelement an das übergeordnete Fenster gesendet, wenn der Anwender [*UMSCHALT*][*Tab*] drückt, während das Dialogelement den Tastaturfokus hat. Das übergeordnete Fenster kann in die Verarbeitung der Eingabe eingreifen, indem ein von Null verschiedener Wert zurückgegeben wird.
- In das übergeordnete Fenster können Dialogelemente gezeichnet werden. Da Ihre Anwendung die Darstellungsweise steuert, muß das Element nicht unbedingt wie von Borland definiert aussehen, aber es wird sich genau wie in der entsprechenden Klasse definiert verhalten.

BWCC-Dialogfenster

Vier Dialogfenster steuern die Eigenschaften der BWCC-Dialogelemente:

- Borland Schalter-Eigenschaften
- Borland Schaltfeld-Eigenschaften
- Borland Markierungsfeld-Eigenschaften
- Borland Schattierungs-Eigenschaften

Um ein Eigenschaften-Dialogfenster anzuzeigen, doppelklicken Sie auf das Dialogelement, dessen Eigenschaften Sie verändern wollen.

Jedes Dialogfenster hat ein Eingabefeld für den Titel und den ID-Wert. Die Eigenschaften-Dialogfenster für Schalter, Schaltfelder und Markierungsfelder enthalten Attribute für Spaltenformat, Gesperrt, Gruppe, Sichtbar und Rahmen sowie Benachrichtigen und OwnerDraw (siehe oben).

Nachfolgend werden die Charakteristika der Eigenschaften-Dialogfenster erläutert.

Dialogfenster für Schalter-Eigenschaften

In diesem Dialogfenster können Sie einen von drei Schaltertypen auswählen: Aktionsschalter, Standardaktionsschalter oder Bitmap.

Aktionsschalter und Standardaktionsschalter
Die Standardeinstellung ist Aktionsschalter. Ein Standardaktionsschalter hat einen fett gezeichneten Rahmen, um dem Anwender anzuzeigen, daß es sich um den *voreingestellten Schalter* handelt; er wird aktiviert, wenn der Anwender die Eingabetaste drückt. Dies ist nicht der Fall, wenn der Tastaturfokus auf einem Texteingabe-Element liegt, für das einWantReturnFlag gesetzt ist).

Wenn Sie in Ihr Dialogfenster ein Dialogelement neu einfügen, lautet der Text Button und erhält den nächsten freien ID-Wert. Um den Schalter in einen von Borland vordefinierten Schalter zu ändern, ändern Sie den ID-Wert in einen der nachfolgenden Werte:

Tabelle A.1 : Vordefinierte BWCC-Schalterelemente

ID Name	ID-Wert	Typ	Symbol
IDOK	1	OK	Grünes Häkchen
IDCANCEL	2	Cancel	Rotes Kreuz
IDABORT	3	Abort	Panik-Schalter
IDRETRY	4	Retry	Einarmiger Bandit
IDIGNORE	5	Ignore	Geschwindigkeitsbegrenzungs-Schild
IDYES	6	Ja	Grünes Häkchen
IDNO	7	Nein	Roter durchgestrichener Kreis
IDHELP	998	Hilfe	Blaues Fragezeichen

Bitmap
Wenn Sie die Option Bitmap auswählen, können Sie eine Bitmap-Grafik in den Schalter einsetzen. Um eine Bitmap-Grafik einzulesen, gehen Sie folgendermaßen vor:

1. Fügen Sie einen BWCC-Aktionsschalter in Ihr Dialogfenster ein, und notieren Sie seinen ID-Wert.
2. Wechseln Sie in den Bitmap-Editor, und erstellen Sie dort eine Grafik.
3. Wählen Sie im Bitmap-Editor Ressource/Umbenennen, um das Dialogfenster Ressource/Umbenennen aufzurufen. Dann stehen Ihnen folgende Möglichkeiten:
 - Geben Sie im Eingabefeld *Neuer Name* einen ganzzahligen Wert ein, der dem ID-Wert Ihres Aktionsschalters entspricht, und einen entsprechenden Offset-Wert aus Tabelle
 - Benennen Sie die Bitmap-Grafik um, und weisen Sie ihr anschließend einen Bezeichner zu, dessen Wert dem ID-Wert Ihres Aktionsschalters entspricht, und einen entsprechenden Offset-Wert aus Tabelle A.2.

Die Bitmap-Grafik erscheint erst dann im Dialog-Editor, wenn Sie den Bitmap-Editor schließen.

4. Schließen Sie den Bitmap-Editor.
5. Kehren Sie zum Dialog-Editor zurück. Falls die Bitmap-Grafik nicht sofort innerhalb des BWCC-Aktionsschalters erscheint, passen Sie dessen Größe an die Größe der Grafik an. Die Grafik wird dann angezeigt werden.

Tabelle A.2 : Bitmap Offset-Werte

Status des Aktionsschalters	Offset für VGA+	Offset für EGA/monochrom
Standard	1000	2000
Gedrückt	3000	4000
Hält Tastaturfokus	5000	6000

Beispiel: Um eine Bitmap-Grafik darzustellen, die sich auf einen Schalter bezieht, der den Tastaturfokus hat und dessen ID-Wert 276 ist, geben Sie 5276 für einen VGA-Bildschirm und 6276 für einen EGA-Bildschirm ein.

Dialogfenster für Schaltfeld-Eigenschaften

In diesem Dialogfenster können Sie zwischen zwei Schaltfeld-Typen wählen:

- Schaltfeld: Ein- und Ausschalten sind nicht voreingestellt. Die Anwendung muß die Funktion **CheckRadioButton** aufrufen, um eine BM_SETCHECK-Botschaft zu senden, die das gewählte Schaltfeld hervorhebt und alle anderen Schaltfelder der gleichen Gruppe ausschaltet.

Automatisches Schaltfeld: Ein Schaltfeld, das Teil einer Gruppe ist. Wenn ein Schaltfeld der Gruppe eingeschaltet wird, werden alle anderen Schalter der Gruppe unabhängig vom Anwendungsprogramm ausgeschaltet. Dies ist die Standardeinstellung.

Dialogfenster für Markierungsfeld-Eigenschaften

In diesem Dialogfenster können Sie zwischen vier Markierungsfeld-Typen wählen:

- Markierungsfeld: Dieses Feld wird nicht automatisch markiert. Das Anwendungsprogramm muß die Funktion **CheckDlgButton** aufrufen, um eine BM_SETCHECK-Botschaft zu senden, die das ausgewählte Markierungsfeld setzt.
- Automatisches Markierungsfeld: Ein Markierungsfeld, das seinen Zustand automatisch ändert, wenn es angeklickt wird. Dies ist die Standardeinstellung.
- Dreistufiges Markierungsfeld: Dieses Feld wird nicht automatisch markiert. Das Anwendungsprogramm muß die Funktion CheckDlgButton aufrufen, um eine BM_SETCHECK-Botschaft zu senden, die das ausgewählte Markierungsfeld setzt.

Die Einstellungen des Markierungsfeldes sind »On«, »Off« und »unbestimmt« (wird als Schachbrettmuster angezeigt). Die Bedeutung von »unbestimmt« wird über das Anwendungsprogramm definiert.

- Automatisches dreistufiges Markierungsfeld: Ein Markierungsfeld, das seinen Zustand automatisch ändert, wenn es angeklickt wird.

Dialogfenster für Schattierungs-Eigenschaften

Über dieses Dialogfenster werden die Attribute für die Dialogelemente Gruppenfeld, Horizontale Tiefe und Vertikale Tiefe festgelegt.

Diese können Sie mit Hilfe der Schaltfelder wie folgt ändern:

- Gruppenfeld in Raised Shade: Gruppenfelder und Raised Shade werden eingesetzt, um Dialogelemente mit verwandten Funktionen, wie z.B. Markierungs- oder Schaltfelder, zu gruppieren. Gruppenfelder erscheinen tiefer, Raised Shade stehen hervor.
- Horizontale Tiefe in horizontalen Grat, Vertikale Tiefe in vertikalen Grat: Tiefen werden üblicherweise für den Hintergrund des Dialogfensters oder für Raised Shades eingesetzt, Grate eignen sich gut für Gruppenfelder.

Dialogfenster für statischen Text

In diesem Dialogfenster können Sie statischen Text definieren. Außerdem können diesem Text Attribute zugewiesen werden.

Neben den Standardattributen (Gesperrt, Gruppe, Sichtbar) hat statischer Text zwei weitere Attribute:

- Falls die Rahmen-Option ausgewählt wurde, wird der statische Text von einem Standard-Windows-Rahmen umgeben, der in der aktuellen Rahmenfarbe gehalten ist (Windows-Systemsteuerung).
- Wenn die Option Nicht unterstrichen gewählt wurde, erscheint ein Und-Zeichen (&) vor dem sonst unterstrichenen Buchstaben.

Die Standardattribute werden an anderer Stelle erläutert. Im Unterschied zu normalem Text wird *jeder* statische Text auf dem grauen Standard-BWCC-Hintergrund dargestellt.

Anpassen bestehender Anwendungen an BWCC

Mit dem Resource Workshop können Sie bestehende Windows-Anwendungen mit benutzerdefinierten Dialogelementen versehen. Die Bearbeitung erfolgt in zwei Schritten:

1. Ändern Sie Ihre WIN.INI-Datei so, daß die BWCC-Bibliothek beim Starten von Windows mitgeladen wird.

Die folgenden Abschnitte beschreiben diese Schritte genauer.

2. Bearbeiten Sie die Anwendung mit dem Resource Workshop, um die einzelnen Elemente Ihrer Benutzeroberfläche, wie z.B. Dialogfenster, Menüs und Symbole, zu ändern.

BWCC immer in Windows laden

Die BWCC-Bibliothek, die die benutzerdefinierten Dialogelemente unterstützt, muß geladen sein, bevor eine Anwendung auf benutzerdefinierte Dialogelemente zugreifen kann.

Editieren Sie die WIN.INI-Datei, die sich im Windows-Verzeichnis befindet (z.B. C:\WINDOWS), so daß Windows beim Start die Datei LOADBWCC.EXE in den Hauptspeicher lädt. Das Installationsprogramm stellt LOADBWCC.EXE in das Verzeichnis, in dem der Compiler die ausführbaren Programme ablegt, und fügt dieses Verzeichnis dem DOS-Suchpfad (PATH) hinzu.

Stellen Sie LOADBWCC.EXE an den Anfang der Dateiliste, die nach dem Schlüsselwort »LOAD=« aufgeführt ist. LOADBWCC.EXE muß das erste Programm in dieser Liste sein, damit BWCC bereits in den Speicher geladen ist, bevor ein möglicherweise modifiziertes Programm ausgeführt wird.

Die LOAD-Anweisung in Ihrer WIN.INI-Datei lautet beispielsweise:

```
LOAD=NWPOPUP.EXE AD.EXE
```

In diesem Fall muß diese Anweisung folgendermaßen geändert werden:

```
LOAD=loadbwcc.exe NWPOPUP.EXE AD.EXE
```

BWCC in C- und C++-Programmen

Falls Sie die IDE von Borland C++ verwenden, klicken Sie BWCC an, wenn Sie eine Projekt-Zieldatei erstellen. Außerdem müssen Sie

```
BWCCGetVersion();
```

in WinMain einfügen.

Falls Sie die IDE nicht verwenden, sind die folgenden Anweisungen zu beachten:

- Fügen Sie Ihrer .C- oder .CPP-Datei ein **#include** bezüglich BWCC.H hinzu.
- Fügen Sie Ihrem C- oder C++-IDE-Projekt die Datei BWCC.LIB hinzu, oder fügen Sie sie in die TLINK-Bibliothek vor IMPORT.LIB ein.
- Fügen Sie in WinMain folgende Zeile ein:

```
BWCCGetVersion();
```

Editierung bestehender Ressourcen

Dieser Abschnitt enthält einige wichtige Aspekte, die bei der Editierung von Ressourcen zu beachten sind. Tastenkürzel

- **Tastenkürzel** Beim Hinzufügen eines Tastenkürzels ist darauf zu achten, daß derselbe ID-Wert zurückgegeben wird wie der ID-Wert des entsprechenden Menüpunktes. Anderfalls führt das Tastenkürzel entweder den falschen Befehl aus oder es erfolgt keine Reaktion.
- **Bitmaps, Cursor, Symbole** Sie können Bitmaps, Cursor-Eigenschaften und Symbole verändern. Löschen Sie jedoch auf keinen Fall Bitmaps, Cursor und Symbole, und versuchen Sie auch nicht, neue hinzuzufügen. In den meisten Fällen wird sie die Anwendung nicht verwenden können.
- **Dialogfenster** Sie können die Elemente innerhalb eines Dialogfensters neu anordnen und statt der Standard-Dialogelemente die entsprechenden benutzerdefinierten Dialogelemente einsetzen.

 Achten Sie darauf, daß während Ihrer Änderung der Typ Ihres Dialogelements mit dem ID-Wert übereinstimmt. Wenn das Dialogelement 100 beispielsweise ein Markierungsfeld ist, ändern Sie es nicht in ein Schaltfeld, denn die Anwendung wird das Dialogelement weiterhin als Markierungsfeld behandeln.

 In den meisten Fällen können Sie Dialogelemente entfernen, die nicht direkt mit der Funktionalität Ihrer Anwendung zusammenhängen. Beispielsweise können Sie in der Regel einen Titel - ein Textelement, das keine Auswirkung auf die Ausführung Ihres Programms hat - entfernen. Es empfiehlt sich jedoch nicht, ein Eingabefeld zu entfernen, da die Ausführung beeinträchtigt wird.

 Fügen Sie dem Dialogfenster keine neuen Dialogelemente hinzu; die Anwendung wird Sie nicht nutzen können.
- **Menüs** In den meisten Anwendungen spielt es keine Rolle, in welcher Reihenfolge die einzelnen Punkte in den Menüs aufgeführt sind. Verschieben Sie aber keine Menübefehle von einem Menü in ein anderes Menü. Die Anwendung wird in diesem Fall nicht mehr in der Lage sein, die kontextsensitive Hilfe aufzurufen oder einzelne Menüpunkte zu aktivieren oder zu deaktivieren.

 Ändern Sie niemals die Reihenfolge der Menüs in der Menüleiste. Verschieben Sie beispielsweise das Menü *Datei* nicht von links nach rechts auf das Menü Bearbeiten.
- **String-Tabellen** Bei der Änderung bestehender String-Tabellen ist Vorsicht angebracht. Manche Programme laden die Strings in Puffer mit festgelegter Größe, und das Hinzufügen von Text zu einem bestehenden String könnte möglicherweise zu einem Puffer-Überlauf führen.

 Fügen Sie keine neuen Strings hinzu; die Anwendung wird sie nicht nutzen können.

A N H A N G C

Vorkompilierte Header-Dateien

Borland C++ ist in der Lage, vorkompilierte Header für Projekte zu erzeugen und diese dann der Reihe nach zu verwenden. Vorkompilierte Header-Dateien erhöhen die Kompilierungsgeschwindigkeit spürbar.

Arbeitsweise vorkompilierter Header-Dateien

Wenn Sie umfangreiche C- oder C++ Programme kompilieren, verbringt der Compiler die Hälfte der Zeit mit dem Zerlegen (parse) der Header-Dateien. Dabei trägt der Compiler Deklarationen und Definitionen in seine Symboltabelle ein. Wenn zehn Ihrer Quelldateien dieselbe Header-Datei benutzen, wird diese auch zehnmal durchsucht und jedesmal wieder dieselbe Symboltabelle erzeugt.

Vorkompilierte Header-Dateien verkürzen diesen Prozeß spürbar. Während der Kompilierung wird vom Compiler BCC.EXE auf der Festplatte ein binäres Abbild der Symboltabelle unter dem Namen BCDEF.CSM angelegt (Für den 32-Bit-Compiler BCC32.EXE trägt diese Datei den Namen BC32DEF.CSM). Dieses Abbild wird im Unterverzeichnis angelegt, in dem sich der jeweilige Compiler befindet.

Wenn später dieselbe Quelldatei (oder eine andere Quelldatei mit derselben Header-Datei) erneut kompiliert wird, lädt der Compiler BCDEF.CSM das Abbild wieder von der Platte, anstatt alle Header-Dateien erneut zerlegen zu müssen. Das direkte Einlesen dieser Symboltabelle von der Festplatte ist nahezu zehnmal schneller als das Zerlegen des in den Header-Dateien enthaltenen Textes.

Vorkompilierte Header-Dateien werden nur verwendet, wenn bei der zweiten Kompilierung eine oder mehrere gleiche Header-Dateien verwen-

det werden, die bereits für die erste Kompilierung angegeben waren, und die Compileroptionen, definierten Makros, u.s.w. ebenfalls identisch sind.

Stellt Borland C++ bei der Kompilierung einer Header-Datei fest, daß die ersten **#include**-Anweisungen identisch mit denen der vorausgehenden Kompilierung sind (bei gleicher oder anderer Quelldatei), wird für diese **#include**-Dateien das binäre Abbild geladen, während die verbleibenden Dateien zerlegt werden.

Der Einsatz vorkompilierter Header-Dateien steht unter dem Motto "Alles oder Nichts": Eine vorkompilierte Header-Datei wird nicht aktualisiert, wenn die Kompilierung einer der eingeschlossenen Header-Dateien nicht erfolgreich verlief.

Nachteile vorkompilierter Header-Dateien

Bei der Verwendung vorkompilierter Header-Dateien kann die Datei BCDEF.CSM sehr umfangreich werden, da sie die binären Abbilder der Symboltabellen aller in die Quellen einzubindenden Include-Dateien enthält. Falls Sie nicht genügend freie Festplattenkapazität besitzen, wird die Warnung ausgegeben, daß der Schreibzugriff auf die Abbild-Datei fehlschlug. In diesem Falle muß Platz auf der Festplatte geschaffen und die Kompilierung nochmals gestartet werden. Weitere Informationen zur Reduzierung der Größe der Datei BCDEF.CSM finden Sie im Abschnitt "Optimierung vorkompilierter Header-Dateien".

Falls Sie große Makros innerhalb einer Make-Datei und gleichzeitig vorkompilierte Header-Dateien benutzen, dann wird die Größe dieser Makros folgendermaßen begrenzt: 4 KB für 16-Bit-Anwendungen und 16 KB für 32-Bit-Anwendungen.

Enthält eine Header-Datei Programmcode, dann kann sie nicht "vorkompiliert" werden. Befinden sich beispielsweise Definitionen von C++ Klassen in den Header-Dateien, dann sollten Sie darauf achten, daß nur Inline-Elementfunktionen darin definiert sind. Achten Sie auf Warnungen wie "Functions containing for are not expanded inline".

Vorkompilierte Header-Dateien verwenden

Die Benutzung vorkompilierter Header-Dateien kann wie folgt gesteuert werden:

- In der IDE steht dazu das Dialogfenster Projekt Optionen zur Verfügung. Die IDE legt den Namen für die vorkompilierte Header-Datei auf der Grundlage des Projektnamens fest und erzeugt eine Datei mit dem Namen *PROJECT*.CSM.

- In der Kommandozeile können Sie die Optionen **-H**, **-H=*Dateiname***, **-Hc**, **-H"Dateiname"** und **-Hu** verwenden.
- Innerhalb Ihres Quelltextes solten Sie die Pragmen *hdrfile* und *hdrstop* einsetzen (siehe Kapitel 5 des *Programmierhandbuches*)

Die Benennung vorkompilierter Header-Dateien

Der Compiler benutzt nur eine Datei für alle vorkompilierten Header-Dateien. Per Voreinstellung trägt diese Datei den Namen BCDEF.CSM. Mit der Kommandozeilen-Option **-H=*Dateiname*** oder der Direktive #pragma **hdrfile** können Sie einen Namen explizit vergeben.

Vorsicht!

Unter Umständen stellen Sie fest, daß Ihre .CSM-Datei kleiner ist als erwartet. In diesem Fall hatte der Compiler beim Schreiben der .CSM-Datei nicht genügend Plattenspeicher zur Verfügung. Der Compiler löscht dann die .CSM-Datei, um Platz für die .OBJ-Datei zu schaffen und legt danach eine neue (und darum kleinere) .CSM-Datei an. Wenn dieser Fall eintritt, müssen Sie vor dem Kompilieren den freien Speicherbereich vergrößern.

Erforderliche Übereinstimmungen

Folgenden Bedingungen müssen erfüllt sein, um eine vorkompilierte Header-Datei benutzen zu können:

Die zweite oder jede weitere Quelldatei muß:

- dieselben Include-Dateien in derselben Reihenfolge einbinden.
- dieselben Makros mit denselben Werten definiert haben.
- in derselben Sprache programmiert sein (C oder C++).
- Header-Dateien mit denselben Zeitangaben (für die Erstellung) verwenden; diese Header-Dateien können entweder direkt oder indirekt einbezogen werden.

Außerdem muß die nachfolgende Quelldatei mit denselben Einstellungen für folgende Optionen kompiliert werden (Wenn Sie die erste Datei beispielsweise mit dem Speichermodell Small kompiliert haben, dann muß die zweite ebenfalls mit diesem Speichermodell kompiliert worden sein):

- Speichermodell, einschließlich SS != DS (**-mx**);
- Voranstellung eines Unterstrich bei externen Symbolen (**-u**);
- maximale Bezeichnerlänge (**-lL**);
- Ziel DOS (Voreinstellung) oder Windows (**-W** oder **Wx**);
- DOS-Overlay-kompatibler Quelltext(**-Y**);
- Virtuelle Tabellenkontrolle (**-Vx** und **-Vmx**);
- Inline Erweiterung des Funktionscodes (**-Oi**) ;
- String-Literale im Codesegment (**-dc,** - nur in der 16-Bit-Version) ;

- Debugger-Informationen (**-v, -vi** und **-R**);
- Far-Variablen (**-Fx**);
- Wortweise Ausrichtung (**-a**);
- Pascal Aufrufkonvention (**-p**) ;
- Behandlung von Aufzählungstypen als Integer (**-b**) ;
- Voreinstellung für **char** als **unsigned** (**-K**) ;
- ANSI-Sprachstandard (**-A**);
- C++ Kompilierung (**-P**)

Optimierung vorkompilierter Header-Dateien

Beachten Sie folgende Regeln, um die Verwendung vorkompilierten Header-Dateien in Borland C++ so effizient wie möglich zu gestalten:

- Ordnen Sie die Header-Dateien in allen Quelltextdateien in derselben Reihenfolge an.
- Stellen Sie die größte Headerdatei an den Anfang.
- Versehen Sie die BCDEF.CSM mit häufig verwendeten Anfangssequenzen der Header-Dateien
- Verwenden Sie #pragma **hdrstop**, um die Liste der Header-Dateien an geeigneten Stellen abzuschließen. Dadurch sieht die Liste der Header-Dateien für den Compiler in verschiedenen Quelltexten ähnlich aus. #pragma **hdrstop** wird im Kapitel 5 des *Programmierhandbuches* eingehender erläutert.

Im folgenden Beispiel sind die beiden Quelltextdateien ASOURCE.C und BSOURCE.C dargestellt, die jeweils die Dateien windows.h und myhdr.h einbinden:

ASOURCE.C:

```
#include  <windows.h>
#include "myhdr.h"
#include "xxx.h"
<...>
```

BSOURCE.C:

```
#include "zz.h"
#include <string.h>
#include "myhdr.h"
#include  <windows.h>
<...>
```

Sie sollten nun die Include-Anweisungen zu Beginn von BSOURCE.C umgruppieren:

Verbesserte Version von BSOURCE.C:

```
#include  <windows.h>
#include "myhdr.h"
#include "zz.h"
#include <string.h>
<...>
```

Beachten Sie, daß windows.h und myhdr.h in BSOURCE.C in derselben Reihenfolge angegeben sind wie in ASOURCE.C. Sie können auch eine neue Quelltextdatei mit dem Namen PREFIX.C anlegen, die nur die Include-Anweisungen für die einzubindenden Header-Dateien enthält:

PREFIX.C

```
#include <windows.h>
#include "myhdr.h"
```

Wenn Sie die Datei PREFIX.C zuerst kompilieren (oder in den beiden Dateien ASOURCE.C und BSOURCE.C nach der Anweisung `#include "myhdr.h"` ein `#pragma hdrstop` einfügen), dann sind nach der ersten Kompilierung von PREFIX.C sowohl ASOURCE.C als auch BSOURCE.C in der Lage, die von PREFIX.C angelegte Symboltabelle zu laden. Der Compiler muß danach nur xxx.h für ASOURCE.C und zz.h bzw. string.h für BSOURCE.C zerlegen.

A N H A N G D

Verwendung von EasyWin

EasyWin ist ein in Borland C++ zur Verfügung stehendes Hilfsmittel, mit dem Sie solche Standard-DOS-Anwendungen kompilieren können, in denen die Ein- und Ausgabe im traditionellen "TTY-Stil" ausgeführt ist. Diese können als echte WINDOWS-Programme ausgeführt werden. Für die Ausführung eines DOS-Programms unter Windows mit Hilfe von EasyWin sind keinerlei Änderungen in diesem DOS-Programm erforderlich.

Einfacher Übergang vom Textbildschirm zu WINDOWS

Wollen Sie Textbildschirm-basierte Anwendungen umwandeln, die Standarddateien oder iostream-Funktionen benutzen, dann müssen Sie in TargetExpert in der IDE überprüfen, ob Sie den richtigen EasyWin-Zieltyp eingestellt haben. Wenn Sie den Kommandozeilen-Compiler verwenden, dann sollten Sie den Compilerschalter (**-W**) angeben. Borland C++ berücksichtigt dann, daß Ihr Programm keine Funktion *WinMain* enthält (was normalerweise für WINDOWS-Anwendungen unbedingt erforderlich ist), und linkt die EasyWin-Bibliothek mit hinzu. Wenn Sie das konvertierte Programm unter WINDOWS ausführen, dann wird ein Standardfenster erzeugt. Das Programm führt die Ein- und Ausgaben in diesem Fenster genauso aus, als würde es sich hierbei um den Standardbildschirm handeln.

Sehen Sie sich folgendes C-Beispielprogramm an:

```
#include <stdio.h>
main()
{
   printf("Hallo, Welt\n");
   return 0;keepfollowing_
}
```

In C++ könnten Sie schreiben:

```
#include <iostream.h>
main()
{
   count << "Hallo, Welt\n";
   return 0;keepfollowing_
}
```

Das EasyWin-Fenster kann jedesmal dann verwendet werden, wenn eine Eingabe von einem TTY-Gerät angenommen werden soll oder eine Ausgabe an ein TTY-Gerät erforderlich ist. Das bedeutet, daß außer den stdin- und stdout- auch noch die stderr-, stdaux- und cerr-"Geräte " mit diesem Fenster verbunden sind.

InitEasyWin()

Die eigentliche Aufgabe von EasyWin besteht darin, DOS-Anwendungen einfach und schnell in WINDOWS-Programme umzuwandeln. Es gibt aber Situationen, in denen Sie vielleicht EasyWin aus einem "echten" WINDOWS-Programm heraus aufrufen wollen. Sie können zum Beispiel die *printf*-Funktionen in Ihren Programmcode einbinden, um sich die Fehlersuche in Ihrem WINDOWS-Programm zu erleichtern.

Wollen Sie EasyWin aus einem WINDOWS-Programm heraus nutzen, dann müssen Sie die Funktion *_InitEasyWin* aufrufen, bevor irgendeine Standardein- oder ausgabe gestartet wird.

Betrachten Sie folgendes Beispiel:

```
#include <windows.h>
#include <stdio.h>
#pragma argsused
int PASCAL WinMain(HANDLE hInstance, HANDLE hPrevInstance,
                   LPSTR lpszCmdLine, int cmdShow)
{
   _InitEasyWin();
   /* Normale Fenstereinstellung */
   printf("Hallo, Welt\n");
   return 0;
}
```

Den Prototyp für *_InitEasyWin* finden Sie in stdio.h und iostream.h.

Weitere Funktionen

Zur Erleichterung Ihrer Arbeit enthält EasyWin fünf weitere Funktionen. Über diese können Sie die X- und Y-Fensterkoordinaten für die Ein- und Ausgabe angeben, das Fenster löschen oder den Fensterbereich von der aktuellen Position bis zum Ende der Zeile löschen:

Eine Beschreibung der in EasyWin-Programmen zur Verfügung stehenden Funktionen finden Sie im *Referenzhandbuch*, Kapitel 3.

clreol	(conio.h)	*wherex*	(conio.h)
clrscr	(conio.h)	*wherey*	(conio.h)
gotoxy	(conio.h)		

Diese Funktionen tragen den gleichen Namen (und besitzen den gleichen Verwendungszweck) wie die jeweilige Funktion in der Header-Datei `conio.h`. Die Klassen aus der `constrea.h` bieten Ihnen die conio-Funktionalität für die Verwendung mit C++ Streams. Eine vollständige Erläuterung zu den *constreams* und *iostreams* finden Sie im *Programmierhandbuch*, Kapitel 6.

Folgende Routinen können in EasyWin-Programmen, nicht aber in 16-Bit-Windows-Programmen genutzt werden. Sie wurden zur Verfügung gestellt, um die Portierung von existierendem Code nach 16-Bit-Windows-Anwendungen zu erleichtern.

fgetchar	(stdio.h)	*printf*	(stdio.h)
getch	(stdio.h)	*putch*	(conio.h)
getchar	(stdio.h)	*putchar*	(stdio.h)
getche	(stdio.h)	*puts*	(stdio.h)
gets	(stdio.h)	*scanf*	(stdio.h)
kbhit	(conio.h)	*vprintf*	(stdio.h)
perror	(errno.h)	*vscanf*	(stdio.h)

Index

!

!elif (MAKE-Direktive) *251*, *253*
!else (MAKE-Direktive) *251*, *253*
!endif (MAKE-Direktive) *251*, *253*
!error (MAKE-Direktive) *251*
 Beschreibung *252*
!if (MAKE-Direktive) *251*, *253*
!ifdef (MAKE-Direktive) *251*, *253*
!ifndef (MAKE-Direktive) *251*, *253*
!include (MAKE-Direktive) *251*
 Beschreibung *254*
!message (MAKE-Direktive) *252*
 Beschreibung *255*
!undef (MAKE-Direktive) *252*
 Beschreibung *256*
 ID
 Quelle *353*
 WM_ENTERIDLE Nachrichten *291*
#define *107*
#define Anweisungen *279*
#pragma intrinsic *107*
#pragma warn *109*
$*, MAKE-Makro
 Kompatibilität zu NMAKE *242*
$d, MAKE-Makro
 Kompatibilität zu NMAKE *242*
& (MAKE-Befehls-Präfix) *247*
* (TLIB-Aktionssymbol) *235*
+ (TLIB-Aktionssymbol) *235*
- (TLIB-Aktionssymbol) *235*
-* und *- (TLIB-Aktionssymbole) *235*
-+ und +- (TLIB-Aktionssymbole) *235*
-?, MAKE-Option *240*
-a, MAKE-Option *240*, *252*
-c, MAKE-Option *240*
-D, MAKE-Option *240*
-e, MAKE-Option *240*
-f, MAKE-Option *240*
-h, MAKE-Option *240*
-I, MAKE-Option *240*
-K, MAKE-Option *240*
-m, MAKE-Option *240*
-N, MAKE-Option *240*
-p, MAKE-Option *240*
-q, MAKE-Option *240*
-r, MAKE-Option *240*
 BUILTINS.MAK und *239*
-s, MAKE-Option *240*, *241*
-U, MAKE-Option *240*
-W, MAKE-Option *240*
 Voreinstellungen verändern mit *241*
.autodepend (MAKE-Direktive) *251*, *252*
.BMP-Datei *263*
.CKB-Dateien
 Beschreibung *36*
.CUR Datei *267*
.CUR-Datei *263*
.DLG-Dateien *262*
.DRV-Datei *262*
.FNT Datei *267*
.FNT-Datei *263*
.HLP-Dateien
 erzeugen *127*
.HPJ-Dateien
 erzeugen *127*
.ICO Datei *267*
.ICO-Datei *263*
.ignore (MAKE-Direktive) *251*
.noautodepend (MAKE-Direktive) *252*
.noIgnore (MAKE-Direktive) *252*
.nosilent (MAKE-Direktive) *252*
.noswap (MAKE-Direktive) *252*
.path.ext (MAKE-Direktive) *252*
 Beschreibung *255*
.precious (MAKE-Direktive) *252*
 Beschreibung *255*
.RC Datei *267*
.RC Dateityp *267*
.rcdata-Datei *261*
.RES Datei *267*
.RTF-Dateien
 erzeugen *127*
.silent (MAKE-Direktive) *252*
.suffixes (MAKE-Direktive) *252*
 Beschreibung *256*
.swap (MAKE-Direktive) *252*

.VBX Dialogelemente und Ressource Workshop *298*
/0 (TLIB-Option) *232*
/a (TLINK-Option)
 Seitenausrichtung *206*
 Typ des Anwendungsprogramms) *206*
/B (TLINK32-Option Basisadresse) *207*
/C
 TLINK-Option (Groß-/Kleinschreibung beachten) *207*
/C (TLIB-Option) *232*
/c (TLINK32-Option (Groß-/Kleinschreibung beachten) *207*
/c, TLINK-Option (Groß-/Kleinschreibung beachten) *207*
/E (TLIB-Option) *232*
/e, TLINK-Option (Erweiterte Wörterbücher$e ignorieren) *207*
/E, TLINK-Option (Erweiterte Wörterbücher) *207*
/E, TLINK32-Option (Maximal erlaubteFehlerzahl) *207*
/f, TLINK-Option (Verhindern von Far-Optimierungen *207*
/Gx, TLINK-Option *208*
/i, TLINK-Option (Nichtinitialisierte Endsegmente) *208*
/l, TLINK-Option (Zeilennummern *208*
/n, TLINK-Option (Standardbibliotheken ignorieren) *208*
/o
 TLINK-Option (Overlays) *209*
/P (TLIB-Option) *232*
/S, TLINK-Option (Stack-Größe) *210*
/t, TLINK-Option (COM-Zieldatei für DOS) *210*
/Td, TLINK-Option (DOS-Zieldatei) *211*
/Tp, TLINK32-Option (Zieldateioption) *211*
/v, TLINK-Option (Debug-Informationen) *211*
/w, TLINK-Option (Freigabe/Unterdrückung von Warnungen) *211*
/ye, TLINK-Option (Expanded Memory) *212*
3 Zustände automatisch, Schaltertyp *309*
3-D Dialogelemente
 im Ressource Workshop *323*
3-D Effekte
 bei Symbol *386*
32-Bit Compiler
 Prozessor für *99*
80386 Prozessor *76*
80386-Anweisungen
 32-Bit *99*
80486-Anweisungen
 32-Bit *99*
räfix *246*
__cdecl *100*
__fastcall Aufrufkonvention *100*
__pascal Aufrufkonvention *100*
__stdcall Aufrufkonvention) *100*

A

abgekürzte Namen
 DLLs und *229*
Abgerundetes Rechteck *365*
Absolut ausrichten von Dialogelementen an Gitter *300*
Absolute Koordinaten für Dialogfenster *291*
ACBP-Feld *209*
Adresse
 Basis- *207*
Airbrush *363*
 benutzen *363*
 Düsenform bestimmen *375*
 Farben wählen *363*
 Füllmuster bestimmen *375*
 Muster wählen *363*
 Sprayeffekt *364*
Aktionsschalter *487*
Aktionssymbole *234*
Aktionssymoble von TLIB *235*
Aktiver Ausschnitt *367*
Aktualisierung beider Bilder Option *378*
Alias-Namen *104*
Alles (Option) *282*
Alles löschen (Befehl) *163*
Allgemeine Elementeigenschaften Dialogfenster *302*
alloca *107*
Ändern
 Attribute von Haltepunkten *165*
Ändern des Bezeichnernamens *277*
Ändern Taste (Bezeichner Dialogfenster) *281*
Ändern von
 Bitmaps,Cursor,Symbole *491*
 Dialogfenstern *491*
 Menüs *491*
 String-Tabellen *491*
 Tastenkürzeln *490*
Anforderungen

an Hard- und Software *19*
Angehängte Segmente
Nichtinitialisierte *208*
Anhalten eines Programms *154*
Ansehen
Datenelemente *160*
ANSI Violations *109, 110*
ANSI Zeichensatz *318*
Ansicht
CGA Auflösung *381*
EGA Auflösung *381*
VGA Auflösung *381*
Ansicht als Datei (Option) *273*
Ansicht Menü *269*
Ansicht/Texteditor
Befehl *45*
Anwendung
Mehrfachpfad *49*
Anwendungen
Compilierung *43*
erzeugen mit AppExpert *117*
Projekte *43*
Verwaltung *43*
Anwendungsoptionen *125*
Grundoptionen *126*
OLE 2-Optionen *127*
Optionen für die Code-Generierung *128*
Verwaltungsoptionen *129*
Anzahl
anzeigbare Farben (Geräteinformation) *384*
Bits pro Pixel (Geräteinformation) *384*
Farbebenen (Geräteinformation) *384*
Anzahl der Durchläufe
für einen Haltepunkt *166*
Anzeige der Warnungs-Optionen *148*
Anzeige/Horizontal anordnen (Winsight) *171*
Anzeige/Vertikal anordnen (Winsight) *171*
Anzeigen
Deklaration von Symbolen *41*
globale Symbole *40*
Objekthierarchie *39*
Symbole im eigenen Programm *42*
Anzeigen der Farbpalette *368, 371*
Anzeigen von Geräteinformationen *384*
AppExpert *135*
Anwendungen anlegen *117*
erzeugte Dateien *118*
Grundlagen *118*
Optionen *125*
View-Klasse *138*
ASM *76*
Assemblersprache
Debuggen *152*
Attribut "Größe" *209*
Attribut "Kombination" *209*
Attribute *209*
Syntaxhervorhebung und *36*
Auflösung
CGA Standard *381*
EGA/VGA Standard *381*
Aufnehmen von Debug-Informationen (Option) *149*
Aufruf
Editor für benutzerdefinierte Ressource *403*
grafischer Editor *271*
interner Texteditor *271*
Ressourcen Editor mit Ressourcen *282*
Ressourcenn Editor *271*
Aufruf des externen Editors *268*
Aufruf des Paint Editor *380*
Aufruf des Paint-Editor *359*
Aufruf des Ressourcen-Editors
String-Editor *352*
Aufrufargumente (Option) *150*
Aufrufstack
Fenster *154*
Menübefehl *154*
Ausdrücke
Auswerten *158*
Definition *155*
Format *159*
Ausdrücke protokollieren *166*
Ausdrucksauswertung (Dialogfenster) *155*
Ausführbar (Vorgabe) *283*
Ausführbare Dateien *262*
bearbeiten *284*
Ausführungsgeschwindigkeit
Optimierung der *104, 108*
Ausführungsposition *151*
Gehen zur *153, 154*
Ausführungszeiger
Anpassung der Anzeigefarben *166*
Ausgabe
Beobachten der Programm- *150*
Ausrichten der Dialogelemente *305*
Ausrichten eines Bereichs *377*
Ausrichten Kommando *304*
Ausrichten von Dialogelementen an einem Gitter *300*
Ausrichten-Befehl *365*

Ausrichtungsoptionen für Bildlaufleiste *311*
Ausrichtungsoptionen für Dialogelemente *305*
Ausrichtungsoptionen im @BLICKFANG = Eingabefeld *313*
Ausrichtungspalette *305*
Ausschneiden von Bereichen *362*
Ausschnitte (WinSight)
 Auswählen der *172*
Auswahl dehnen Dialogfenster *377*
Auswahl in Rahmennähe-Schaltfeld *296*
Auswahlwerkzeug *296*
Auswerten/Ändern (Dialogfenster) *158*
Autodependency-Information (Projekt-Manager) *45*
Autodependency-Knoten *45*
Automatisch
 Zugriff auf Dateien (Autodependency) *45*
Automatisch (virtuelle Tabellen) *102*
Automatisch Symboldatei erzeugen *381*
Automatische Bezeichnerverwaltung *279*
Automatische Fehlerrückverfolgung *38*
Automatischer Bildlauf *313*
Automatisches dreistufiges Markierungsfeld *489*
Automatisches Markierungsfeld *309, 488*
Automatisches Schaltfeld *488*

B

Backup Dateien erzeugen (Option) *284*
Backupdateien *284*
Basis
 virtuelle *102*
Basisadresse *207*
BCC *76, 77*
BCC.EXE *76, 77, 112*
 TLINK und *202*
BCC32.CFG *77*
BCC32.EXE *76, 112*
 TLINK und *202*
BCCONFIG.BCW *32*
BCCONFIG.BCW) gespeichert. *35*
BCDEF
 CSM *493*
BCDEF.CSM *494*
BCWDEF.DSW *35*
Bearbeiten
 Dialogelemente *300*
Beenden eines Programms *155*
Befehl
 Projekt/Projekt Öffnen *50*
Befehle (MAKE)
 Regeln für *246*
Befehlsregister (IP) *153, 154*
Behandlungsroutine (Handler) *140*
Beispiel
 STYLESHT.IDE *64*
Beispiel einer Header-Datei *279*
Beispiel für Bezeichnerwert Typumwandlung *277*
Beispiele für TLIB *235*
Beispielprojekt, String-Tabelle *356*
Benutzerdefinierte Dialogelemente *318*
 Bibliothek installieren (DLL) *319*
 Darstellung auf dem Bildschirm *294*
 erstellen *318*
benutzerdefinierte Dialogklasse *483*
Benutzerdefinierte Farben
 erstellen *372*
Benutzerdefinierte Ressource *401*
 Änderungen compilieren *404*
 anzeigen lassen *402*
 bearbeiten *403*
 Daten hinzufügen *404*
Benutzerdefinierte Ressourcen *261*
 einbinden ins Projekt *273*
 hinzufügen zum Projekt *402*
Benutzerdefinierte Schaltelemente *310*
Benutzerdefinierter Ressourcen-Typ
 Dialogfenster *405*
Benutzeroberfläche
 erzeugen *118*
Bereich der angezeigten Datenelemente
 Ändern *161*
Bezeichner
 #define *269*
 Ändern des Werts *277*
 Auswirkung auf Laufzeitverhalten *277*
 automatisch verwalten *279*
 beim Umbenennen von Ressourcen *275*
 dem Projekt hinzufügen *280*
 erzeugen beim Ressourcen umbenennen *280*
 erzeugen mit Bezeichner-Dialogfenster *281*
 far Zeiger auf einen char *277*
 Header-Datei *268*
 Identifizierung *277*
 Integerwert *277*
 Konstanten *269*
 löschen *281*
 löschen mit Bezeichner Dialogfenster *281*

Name *277*
Namensgebung *277*
Präfixe *279*
Ressourcen *276*
String-Tabelle *354*
Typ des Werts umwandeln *277*
unbenutzte *281*
Wert *277*
Wert ändern *281*
Bezeichner
speichern *268*
Bezeichner Dialogfenster
Darstellen Gruppe *282*
Verwendung Fenster *282*
Bezeichnerdatei
benennen *278*
erzeugen *278*
Projekt ohne bearbeiten *280*
Typ wählen *278*
Bezeichnerdateien
C-Header-Dateien *278*
Pascal-Units *278*
BGI *48*
Bibliotheken *203*
DPMI-Anwendungen und *203*
erzeugen *235*
Gründe für den Einsatz *231*
Ignorieren durch TLINK *208*
Mathematische *203*
TLINK und *203*
TLINK32 und *203*
Win32-Anwendungen und *203*
Windows-3.x-Anwendungen und *203*
Bibliothken
Objektmodule und *231*
Bild
bearbeiten *383*
eines Symbols erstellen *383*
Bild (Farbvariante) *382*
Bild bearbeiten Befehl *383*
Bildausschnitt vergrößern *362*
Bildausschnitt verkleinern *362*
Bildlaufleiste
Ausrichten im Dialogfenster *311*
Größe ändern *311*
Bildlaufleiste positionieren *310*
Bildlaufleisten
Option Listenfenster auf *311*
Option Selbstzeichnend *311*
Binärdatei *268, 380*
decompilieren *268*
Binärdateien
bearbeiten *284*
Binärformat
Schrift *263*
Binärformat (Ressourcen) *273*
Bitmap *487*
Bitmap Offset-Werte *488*
Bitmap Ressource
Standard *359*
Bitmap-Editor *487*
Bitmap-Grafik *487*
Bitmap-Ressource
Größe festlegen *359*
Standardgröße *359*
Bitmaps *260*
BN_CLICKED Nachricht *309*
Boolescher Ausdruck
Bedingung für Haltepunkt *166*
BORDLG *483*
Borland
CompuServe *21*
Borland C++
Beschreibung *17*
Neuheiten *17*
Borland C++ Version 3.x *55*
Borland Informationssystem
BORIS *21*
Borland Markierungsfeld-Eigenschaften *486*
Borland Schalter-Eigenschaften *486*
Borland Schaltfeld-Eigenschaften *486*
Borland Schattierungs-Eigenschaften *486*
BorlandC++
Neuheiten in Version 4.0 *18*
Botschaft
BBN_GOTABTAB *486*
BBN_GOTATAB *486*
BBN_SETFOCUS *486*
BBN_SETFOCUSMOUSE *486*
BM_SETCHECK *488*
Botschaften
Debug-Informationen über *171*
Botschaften (WinSight)
Abschalten der Verfolgung der *172*
Verfolgen von *176ff*
Botschaften bei WinSight
Auf Drucker ausgeben *176*
Ausgabe auf Hilfsmonitor *176*
Hexadezimalwerte bei *177*
Interpretieren der Parameter *177*

Schreiben in Datei *176*
Botschaften/Optionen *176*
Botschaftsverfolgung *173*
Botschaftswarteschlange (Abschnitt in WINSPCTR.LOG) *192*
BRC
Aufruf *224*
BRC (Ressourcen-Dienstprogramm *224ff*
BRC (Ressourcen-Dienstprogramm) *221*, *223*, *225*
BRC, Optionen *224*
BRCC (Ressourcencompiler von Borland *222*
Aufruf *222*
BRCC (Ressourcencompiler von Borland) *221*, *223*, *225*
Brief
Tastenkürzel für Editor *36*
BRRC, Optionen *222*
BUILDSYM *197*
Erzeugen vom SYM-Dateien mit *197*
BUILTINS.MAK
Beschreibung *239*
Beschreibung von *238*
Inhalt von *239*
BWCC *48*
in C- und C++-Programmen *490*
benutzerdefinierte Elemente für Windows *483*
Dialogklasse *483*
Erweiterungen *485*
BWCCAPI.RW *483*
BWCCGetVersion() *490*
BWCCSTYL.RW *483*

C

C Aufrufkonventionen *100*
C++ Optionen
Elementzeiger- *100*
C++ Warnings *109*
C++-Optionen
virtuelle Tabellen *102*
C++-Tabellen
virtuelle *102*
C-Sprache
Escape-Sequenzen *355*
C0Fx.OBJ *49*
CBS_DISABLENOSCROLL Eigenschaft *318*
CGA Auflösung *381*
CheckRadioButton *488*
ClassExpert *135*
Ansicht *135*
Bearbeitungsausschnitt *136*
Dokumenttypen *138*
Ereignisausschnitt *136*
Klasse hinzufügen *137*
Klassenauschnitt *136*
lokales Bearbeitungsmenü *136*
lokales Ereignismenü *136*
lokales Klassenmenü *136*
Client-Fenster *125*
CM_ (Präfix) *280*
Code
invarianter *107*
toter *106*
Code erzeugen
Übersicht *121*
CODE-Anweisung *213*
COM-Dateien
TLINK und *210*
Compact
Speichermodell *47*
comparision
pointer- *110*
Compiler
Aufrufkonvention für *100*
Optionen *75*, *79*
Speichermodell *81*
Compilieren *268*
Compilierungsfehler *268*
von Projekten *72*
Compilieren von Ressourcen *222*
Compilierfehler
im Programm suchen *37*
Compilierungsfehler Dialogfenster *268*
Compilierzeitfehler *74*
Definition *147*
Häufige Ursachen für *148*
Console
Speichermodell *47*
CONTROL-Anweisungen erzeugen *322*
conversion
pointer- *110*
Copilieren
IDE *75*
CPU-Last
Verminderung *108*
CreateDialog *291*
CSM
CSM-Dateien *493*, *494*, *495*

CSM-Dateien
 Plattenkapazität und *495*
Cursor
 auf Fehler im Programm setzen *37*
CUSTCNTL:RW Datei *318*
CX-Wert *308*
CY-Wert *308*

D

Darstellen Gruppe
 Alles (Option) *282*
 Einzelne Datei (Option) *282*
Darstellung überwachter Ausdrücke (Dialogfenster) *156*
DATA-Anweisung *213*
Datei
 externe *406*
 Verweis im Ressourcenskript *406*
Datei einem Projekt hinzufügen Dialogfenster *274, 278, 404*
Datei-Manager (Windows)
 Quellknoten hinzufügen *51*
Dateien
 Ändern der Datums-/Uhrzeitangaben von *239*
 gruppieren *57*
 Listen- *232*
 Namenserweiterungen *200*
 Namenserweiterungen (angehängt von TLINK) *200*
 Response- *200, 232*
 Syntaxhervorhebung und *36*
 Viewing in der IDE *44*
Dateityp
 .DLL *284*
 .H *278*
 .ICO *379*
 .INC *280*
 .PAS *280*
 .RES *284*
 BMP *267*
 CUR *267, 390*
 FNT *267*
 ICO *267*
 RC *267*
 RES *267*
Daten, benutzerdefinierter Ressource hinzufügen *404*
Datenausrichtung *99*
Datenelemente
 Ansehen *160*
 Modifizieren des Werts *160*
Datums-/Uhrzeitangaben
 Ändern von *239*
Debug-Informationen in OBJs *149*
Debuggen
 Assemblersprache *152*
 DLLs *168*
 Elementfunktionen *152*
 Externe Programme *152*
 im Hard-Modus *169, 170*
 im Soft-Modus *169, 170*
 Map-Dateien *208*
 Neustart beim *155*
 TLINK und das *211*
 WinSight und *171, 173, 175, 177, 179, 181*
 WinSpector und *187*
Debugger
 WinSpector und *187*
DEF-Dateien *227*
Dem Projekt hinzufügen (Befehl) *272*
Dem Projekt hinzufügen Befehl *404*
DESCRIPTION-Anweisung *214*
Destruktor
 Aufräumen des *103*
Detailfenster (WinSight) *173*
DFA zusammen mit Turbo Debugger benutzen *195*
Diagnoseversionen
 CD-Version von Borland C++ *49*
 Diskettenversion von Borland C++ *49*
Diagnostic *49*
Dialog-Editor *483*
 Werkzeugpalette *484*
Dialogdateien *262*
Dialogeditor
 Aufruf *287*
 Benutzerdefinierte Dialogelemente darstellen *319*
 Rückgängig Befegl *308*
 WYSIWYG *322*
 Zwei Dialogfenster anzeigen *320*
Dialogeditor,Vorgaben einstellen *321*
Dialogeinheiten
 Definition *321*
Dialogeinheiten Definition *289*
Dialogelement
 Aktionsschalter *485*
 allgemeine Elementeigenschaften *485*

Gruppenfeld *485*
Horizontale Tiefe *485*
Markierungsfeld *485*
Schaltfelder *485*
Statischer Text *485*
Vertikale Tiefe *485*
Dialogelemente
mit 3-D Effekt *323*
allgemeine Kategorien *293*
als Tabulatorposition definieren *303*
als Tabulatorstops definiert *302*
am Gitter ausrichten *300*
Arbeiten mit *293*
aus Gruppen entfernen *297*
Ausrichten *304, 305*
Ausrichten, Anordnen und Größe verändern *304*
Auswahl durch Tab *302*
Bearbeiten *300*
Bearbeiten rückgängig machen *308*
bei Erstaufruf sichtbar *301*
Benutzerdefinierte *293*
Benutzerdefiniertes Element *318*
Bildlaufleiste hinzufügen zu *301*
Dialogfenster Eigenschaften des Elements *300*
Eigenschaften anzeigen *298*
Eingabefeld *295*
gemeinsame Attribute *301*
gemeinsame Optionen *301*
Gleicher Abstand Ausrichtungsoption *305*
Grau dargestellt *301*
Größe ändern *307*
Größe verändern *299*
Gruppen *301*
Gruppen definieren *303*
Gruppen hinzufügen *297*
Gruppenfeld *295*
Horizontale Ausrichtung Ausrichtungsoption *305*
Horizontale Bildlaufleiste *295*
ID-Wert zuweisen *301*
im Borland Stil *293*
in Spaten anordnen *306*
in Zeilen anordnen *306*
Klasse ändern *302*
Kombinationsfeld *295*
Kopien ins Dialogfenster einfügen *298*
Linke Seite Ausrichtungsoption *305*
Listenfeld *295, 311*
Markieren *296*
markieren von mehreren *296*
Markierungsfeld *295*
Maximale Höhe *308*
Minimale Breite *308*
Nicht verfügbar *301*
Position ändern *296*
Positionieren und skalieren zugleich *299*
Rahmenbreite (Skalierung) *308*
Rechte Seite-Ausrichtungsoption *305*
Reihenfolge ändern *297*
Reihenfolgennummern zuweisen *304*
Schaltfeld *295*
Schwarzer Rahmen *296*
Schwarzes Rechteck *296*
Standard Windows *293*
Statischer Text *295*
Statisches Symbol *296*
Tabulatorposition im Dialogfenster ändern *302*
Taste *295*
Titel hinzufügen *301*
Umordnen *303, 304, 306*
Unverändert Ausrichtungsoption *305*
Ursprüngliche Reihenfolge wiederherstellen *304*
Vertikale Bildlaufleiste *295*
Werkzeugpalette *294*
Zentriert Ausrichtungsoption *305*
Zentriert im Dialog Ausrichtungsoption *305*
Dialogelemente einem Dialogfenster hinzufügen *297*
Dialogelemente verschieben *299*
Dialogelemente, mit Markierungsrahmen hervorgehoben *296*
Dialogelemente, vom SDK Editor erkannt *293*
Dialogelementklasse
BorBtn *485*
BorCheck *485*
BorRadio *485*
BorShade *485*
BorStatic *485*
Dialogfeld
Neues Ziel *46*
Dialogfenster *259*
Attribute *289*
Ausrichtung am Bildschirm *291*
Auswahl dehnen *377*
Bearbeiten rückgängig machen *308*
Benutzedefiniertes Element hinzufügen *319*
bewegen *289*

Bildlaufleiste ausrichten *311*
Datei einem Projekt hinzufügen *274, 278, 404*
Dialogelemente hinzufügen *297*
Dialoggröße *293*
Dialograhmen *290*
Dicken Rahmen zufügen *290*
Eigenschaften ändern *290*
Eigenschaften anzeigen *289*
Eigenschaften der Liste *311*
Eigenschaften des Kombinationsfensters *318*
Elemente in Tabellenform ausrichten *306*
Erweiterte Optionen *50*
erzeugen *287*
Farbe bearbeiten *372*
Fenstereigenschaften *290*
Fenstertyp *290*
Größe ändern *289*
horizontale Bildlaufleiste hinzufügen *291*
Kein Rahmen Option *290*
Keine Idle Nachrichten Option *291*
markieren *289*
Menü hinzufügen *292*
modal *291*
Muster bestimmen *366*
Neue Bitmap Attribute *369*
Neue Bitmap-Attribute *359*
Neue Ressource *272, 403*
Neue Ressourcendatei *380, 390*
Neuer Bezeichner *280*
Neuer ResRessourcen-Typ *402*
Neues Benutzerdefiniertes Element *319*
Neues Symbol *380*
Neues Ziel *46*
Popup Typ *290*
Position von Windows bestimmt *292*
Rahmen *289*
Rahmenart festlegen *290*
Ressource einfügen *275*
Schaltereogenschaften *309*
Schrift ändern *291*
selbstdefinierte Klasse zuweisen *292*
Speicheroptionen für Ressource *276*
Symbol-Attribute *383*
Symbolfeld hinzufügen *291*
Systemmenü zufügen *290*
testen *320*
Titel hinzufügen *290*
Titel Rahmenart Option *290*
Transparente Farbe bestimmen *371*
Überlappend Typ *290*
Unterfenster *290*
verschiebbar *291*
vertikale Bildlaufleiste hinzufügen *291*
Wert des Bezeichners ändern *281*
Zoomfeld hinzufügen *291*
Dialogfenster Compilierungsfehler *268*
Dialogfenster Eigenschaften der Bildlaufleiste *311*
Dialogfenster Eigenschaften des Kombinationsfensters *317*
Dialogfenster Größe der Elemente ändern *307*
Dialogfenster Vorgaben (Dialogeditor) *321*
Dialogfenster, Definition der Auswahl mit Tab *302*
Dialoggröße Dialogfenster *293*
Dialograhmen-Rahmenart Eines Dialogfenster *290*
Dicken Rahmen für Dialogfenster hinzufügen *290*
Discardable (Option) *276*
DLL
Dateien *262*
DLL Datei *319*
DLL Dynamic Link Library Datei *293*
DLLs
abgekürzte Funktionsnamen und *229*
Debuggen in *168*
Importbibliotheken und *230*
Komprimieren von Codesegmenten *209*
und Klassen *229*
verbergen von Export-Funktionen *230*
Dok/View
unterstützen *138*
Dokumenttypen
erstellen *138*
DOS
Overlays und der Linker *209*
DOS-Anwendungen
DS-und SS-Segmente *47*
IDE-Optionen *48*
DOS-Editor
als externer Editor *283*
Dreistufiges Markierungsfeld *488*
DS=SS *80*
Duplikatkopie *54*
Duplizieren-Werkzeug *298*
Durchsehen
in der Windows-IDE *38*
Durchsichtige Farbedes Mauszeigers *391*

Dynamische Linkbibliotheken
Debuggen in *168*

E

EasyWin *499*, *501*
Editierfenster (IDE)
Laden von Quelldateien *45*
Editor
aufrufen *403*
externen wählen *283*
externer *406*
für Ressourcentext *403*
Position bei Fehlern *37*
Syntaxhervorhebung *36*
Tasten *405*
Text hervorheben *36*
Textfarbe *36*
Editor Brief
emulieren *36*
Editor Epsilon
emulieren *36*
Editor starten Taste (bei Compilierungsfehler) *268*
Editorfenster
konfigurieren *35*
EGA-Bildschirm *488*
EGA/VGA Auflösung *381*
Eigenschaften der Bildlaufleiste-Dialogfenster *311*
Eigenschaften der Liste Dialogfenster *311*
Eigenschaften des Kombinationsfensters
Dialogfenster *317*, *318*
Eingabeaufforderungen in String-Tabelle *351*
Eingabemodus im internen Texteditor *272*
Eingebundene Ressourcen *269*
Einträge anzeigen (Option) *270*
Einzelne Datei (Option) *282*
Element duplizieren-Dialogfenster *298*
Elemente
Syntaxhervorhebung und *37*
Elemente ausrichten Dialogfenster *305*
Elemente in Tabellenform
ausrichten-Dialogfenster *306*
Elementfunktionen
Debuggen in *152*
Elementgruppen Werkzeug *303*
Elementtyp-Optionen (Statische
Dialogelemente) *315*
Elementzeiger *100*
Genauigkeit von *100*
kleinste Darstellung von *101*
Elementzeiger (near) *101*
Elementzeiger auf char *101*
Elementzeiger(Einfachvererbung) *101*
Elementzeiger(Mehrfachvererbung) *101*
Ellipse *365*
EM_GETHANDLE Nachrichten *291*
EM_SETHANDLE Nachrichten *291*
Emulation *49*
Entfernen
Haltepunkte *163*
Epilog/Prolog *106*
Epsilon
Tastenkürzel *36*
Ereignisbehandlungsroutine (Event Handler)
hinzufügen *140*
löschen *140*
Ereignisse, Fenster *166*
Errors
Frequent *109*
Ersetzen von Symbolen *235*
Erweiterte Optionen, Dialogfeld *60*
erweitertes Verzeichnis
TLIB und *233*
Erzeugen eines Ressourcen-Typs *402*
Erzeugen neuer Symbole *379*
Escape-Sequencen
wie in C *355*
Exceptions (Optionen) *103*
EXE-Dateien
Debug-Informationen *211*
TLINK und *211*
EXEMAP mit WinSpector benutzen *196*, *197*, *198*
EXEMAP.EXE
Erzeugen von MAP-Dateien mit *196*
EXETYPE-Anweisung *214*
Expanded Memory
TLINK und *212*
EXPORTS-Anweisung *214*
Extended Memory
TLINK und *212*
extern (virtuelle Tabellen) *102*
Externe Datei *406*
Externer Editor *406*
Aufruf *268*
Extrahieren von Symbolen *235*
Extras, Befehl *59*

F

Farbbereich
 invertiert *370*
 transparent *370*
Farbe
 Anzahl wählen *369*
 Farbiger Text *36*
 Hintergrundfarbe *369*
 Palettenindex *368*
 RGB-Einstellung *368*
 Text im Editor *36*
 Vordergrundfarbe *369*
Farbe bearbeiten Dialogfenster *372*
Farbeimer *364*
 benutzen *364*
 Farben wählen *364*
 Gießalgorithmus ändern *364*
 Muster wählen *364*
Farben
 auswählen *368*
 bearbeiten *372*
 gerastert *373*
 Index-Werte *372*
 reservierte *384*
 RGB-Werte ändern *373*
 selbst definieren *372*
 Syntaxhervorhebung und *36*
 vorgegebene von Knoten (Projekt-Manager) *45*
Farben eines Symbols *380*
Farbpalette
 angezeigte Farben ändern *372*
 anzeigen *368, 371*
 benutzerdefinierte Farben *372*
 Farbanzahl wählen *369*
 Farben bearbeiten *372*
 Farbenanzahl Einschränkung *369*
 Hintergrundfarbe benutzen *369*
 Hintergrundfarbe wählen *369*
 invertierte Farbe wählen *369*
 logische *373, 384*
 Palettenindex *372*
 RGB-Einstellung *368*
 selbstdefinierte speichern *373*
 speichern *373, 378*
 transparente Farbe wählen *369*
 verbergen *368, 371*
 Vordergrundfarbe benutzen *369*
 Vordergrundfarbe wählen *369*
Farbpalette anzeigen (Befehl) *371*
Farbpalette verbergen Befehl *371*
Farbpalette:Palettenindex *368*
Farbpaletten (Geräteinformation) *384*
Farbunterstützung unter Windows *382*
Farbvarianten eines Symbols *382*
Fehler
 ansehen *74*
 Arten von *147, 148*
 Begrenzen der angezeigten Anzahl der *148*
 Compilierzeit- (Syntax-) *147*
 Erkennung von *147*
 Häufige ursachen für Compilierzeitfehler *148*
 Häufige Ursachen für Laufzeitfehler *149*
 im Programm suchen *37*
 Logische *149*
 potentielle C++- *111*
 Rückverfolgung im Meldungsfenster *38*
Fehlerbehebung *75*
Fehlermeldungen
 beim Compilieren *74*
 definierte *408*
Fehlermeldungen in String-Tabelle *351*
Fehlermeldungen,schwerwiegende *407*
Fenster
 Client-Fenster *125*
Fenster Überwachter Ausdruck *155*
Fensterausschnitte im Paint-Editor *367*
Fenstereigenschaften Dialogfenster *290*
Fensterhierarchie *172*
Fensterhierarchie (WinSight) *174, 175*
Fensterklassen *171*
 Debug-Information über *171*
Fenstertyp *290*
Festlegen von Knotentypen *50*
Filter
 Windows-Symbolanzeige *40*
Flächen einfärben *364*
Floating Point *49*
Formatieren
 Ausdrücke *159*
 Überwachte Ausdrücke *156*
Formatspezifikatoren
 Ausdrücke *159*
Freihandzeichnung *363*
Frequent Errors *109*
Füllmuster
 bestimmen *375*
Funktion
 CheckDlgButton *488*
 Suchen einer *153*

Funktion überspringen
Definition *150*
Funktion überspringen (Menübefehl) *151*
Funktionen
in Windows durchsehen *38*
inline *494*
Inline Intrinsic- *106*
Inline- *102*

G

GDI (Graphics Device Interface) in WINSPCTR.LOG *193*
Gefüllte Ellipse *366*
Füllmuster bestimmen *375*
Gefüllte Flächen zeichnen *366*
abgerundetes Rechteck *366*
Ellipse *366*
Füllmuster festlegen *366*
Rahmenbreite festlegen *366*
Rahmenfarben wählen *366*
Rechteck *366*
Gefüllte Flächen zeichnen (Werkzeug) *366*
benutzen *366*
Gefüllter abgerundeter Rahmen
Füllmuster bestimmen *375*
Gefüllter Rahmen
Füllmuster bestimmen *375*
Gefülltes abgerundetes Rechteck *366*
Gefülltes Rechteck *366*
Geraden zeichnen *365*
Gerasterte Farben *373*
Geräteinformationen anzeigen Fenster *384*
Gerätetreiberdateien *262*
Geschwindigkeit
Optionen *82*
Gewünscht-Feld (Farben bearbeiten *373*
Gitter auf vergrößertes Fenster Befehl *362*
Gitter zeigen Option
Ausrichten von Dialogelemente *300*
Gitterattribute festlegen-Dialogfenster *300*
Gleicher Abstand (Ausrichtungsoption für Dialogelemente) *305*
Gleichgeordnete Fenster
von Dialogfenstern nicht verdeckbar *291*
Gleitkomma
Mathematische Bibliotheken und *203*
Unterstützung für DOS-Anwendungen *48*
Gleitkomma, Anzeigeoption *157*
global (Templates) *103*
Go to CS:IP, Menübefehl *153*, *154*
Grafischer Editor *271*
Graphics Device Interface (GDI) *401*
Grau dargestellte Dialogelemente *301*
Groß-/Kleinschreibung
TLIB-Option *233*
Groß-/Kleinschreibung (TLINK)
Moduldefinitionsdateien und *207*
Groß/Kleinschreibung in Eingabefeldern *314*
Größe der Elemente ändern-Dialogfenster *299*, *307*
Größe eine Symbols
bestimmen *380*
Größe und Attribute Befehl *383*
Größe und Attribute Befehl (Paint-Editor) *369*
Gruppe
Reihenfolgenkonvention *303*
Gruppen Kommando *303*
Gruppenfeld *489*
GUI
Speichermodell *47*

H

Haltepunkt (Befehl in Mauspalette) *162*
Haltepunkt löschen (Befehl) *163*
Haltepunkt, Dialogfenster *163*
Haltepunkte
Aktivieren *164*
Ändern der Attribute von *165*
Anpassung der Anzeigefarben *166*
Bedingte *166*
Deaktivieren *164*
Definition *162*, *163*, *164*, *165*, *166*
Löschen von *163*
Modifizieren *165*
Setzen in nicht geöffneter Datei *165*
Setzen von *162*
Haltepunkte (Fenster) *163*
Hand benutzen *366*
Hand(Werkzeug) *366*
Hard-Modus *169*, *170*
Hauptfensteroptionen *129*
Grundoptionen *129*
Optionen für den Dialog Client *132*
Optionen für den MDI Client *132*
Optionen für den SDI-Client *130*
Header

vorcompiliert *113*
Header-Datei *268*, *272*, *279*
HEAPSIZE
Syntax *219*
Hexadezimale Darstellung *405*
Hexadezimale Werte in String-Tabellen *355*
Hexadezimales Format *355*
Hexadezimalwerte (WinSight-Option) *177*
hexstring
Datentyp *405*
HG-Hintergrundfarbe in Farbpalette *360*
Hilfe aufrufen *32*
Hilfe-Compiler *47*
Hilfesystem *20*
Hilfsprogramme *58*
zur IDE hinzufügen *60*
Hintergrundfarbe
Zeichnen *360*
Hintergrundfarbe bearbeiten (Befehl) *371*
Hintergrundfarbedes Mauszeigers *391*
Hinzufügen von Modulen *235*
Horizontal teilen (Befehl im Paint-Editor) *367*
Horizontale Ausrichtung (Ausrichtungsoption für Dialogelemente) *305*
Horizontale Tiefe *489*
Huge
Speichermodell *47*

I

ICO (Symboldatei) *379*
ID
Nummer *353*
Wert *353*
Wert für Ressourcen-Typ *402*
ID einer Ressourcen *276*
ID-Wert, Eigenschaft der Dialogelemente *301*
ID-Werte für Aktionsschalter unter Windows *310*
IDC_ (Präfix) *280*
IDE *490*
Compilieren *75*
Eigenschaften *1*
Filter *60*
Hilfsprogramme hinzufügen *62*
konfigurieren *32*
Optionen für *112*
Übersetzer,Viewer und Hilfsprogramme hinzufügen *60*
Verwendung *1*
Verwendung der *27*
Viewer, Übersetzer und Hilfsprogramme *59*
IDE-Fenster
Speicherung von *35*
IDE-Menübefehle *32*
IDE-Optionen
Gleitkomma-Unterstützung *48*
IDE-Statuszeile
Textanzeige in *35*
IDs
String-Tabelle *353*
IDS_ (Präfix) *280*
IMPDEF *227*, *228*, *229*, *230*
Importbibliothek
anpassen einer *230*
Erzeugen einer *228*
Importbibliotheken *227*, *230*
Importieren
Projekte älterer C++ Versionen *55*
IMPORTS-Anweisung *215*
Induktionsvariable *108*
Info Taste (Symbol-Attribute Dialogfenster) *384*
Informationen *20*
Informationen in der Statuszeile *368*
Inline Intrinsic-Funktion *106*
Inline-Funktionen *102*
Inspizieren
Quellcode *164*
Installation *19*
Instanzvariable
hinzufügen *141*
löschen *141*
Intel
Definition des ACBP-Feldes *209*
Interpretation von Werten (WinSight-Option) *177*
intrinsic
#pragma *107*
Intrinsic-Funktion
Inline *106*
Invertierte Farbe *382*
Invertierte Farbe des Mauszeigers *391*
Invertierte Farben
benutzen *370*
einstellen *370*
Farbdarstellung neudefinieren *371*
Voreinstellung *370*

K

Kein Rahmen-Rahmenart eines Dialogfenster *290*
Klasse eines Elements verändern *302*
Klassen
 DLLs und *229*
 erzeugen *135*
Klassenbibliothek *48*
Klassenliste *173*
Klassenliste (WinSight) *172*
Klassenwerte
 Übergabe von *101*
Klicken, Markierungsmethode *296*
Knoten *65*
 Autodependency-Knoten *45*
 compilieren *58*
 kopieren *54*
 Laufzeitknoten (IDE) *45*
 Quellknoten (IDE) *45*
 verschieben *53*
 vorgegebene Farben (Projekt-Manager) *45*
 Zielknoten (IDE) *45*
Knoten, bestimmende
 Eigenschaften bearbeiten *51*
 löschen *51*
 Verweis auf *57*
Knoten,bestimmende
 hinzufügen *50*
Knotenattribute *67*
 Beschreibung *51*
 Knotentyp *52*
 Name *51*
 Schablone *51*
 Übersetzer *52*
Knotentyp
 .C-Knoten *50*
 .CPP-Knoten *50*
 festlegen *50*
 Kein Knoten *50*
Kombinationsfeld *295*
Kombinationsfenster *317*
 Typ-Optionen *317*
Kommandozeilen-Hilfsprogramme
 ausführen *2*
 Beschreibung *1*
Kommandozeilencompiler
 TLINK und *202*
Kommandozeilenoptionen *112, 113, 114, 115, 116*
 von der IDE aus setzen *63*
Kommandozeilenparameter
 Ansehen der übergebenen *154*
 Übergabe ans Programm beim Debuggen *150*
Kommentar-Records
 entfernen von *232*
Kompatibilität
 Rückwärts- *115*
Konfigurationsdateien *77*
Konfigurieren
 IDE-Editor *36*
 Tastenkürzel *36*
Kontaktpunkt *392*
 Pixelkoordinaten bestimmen *392*
Konventionen *19*
Konvertierung
 OEM in ANSI *318*
Koordinatendefinition *308*
Kopieren von Ressourcen *274*

L

Laden
 redundantes *106*
Large
 Speichermodell *47*
Lauf zur Cursorposition (Menübefehl) *153*
Laufzeit
 Fehler (Ursachen) *149*
Laufzeit-Manager *3*
Laufzeitbibliothek *48*
Laufzeitbibliotheken
 Mehrfachpfad *49*
Laufzeitfehler
 Definition *149*
Laufzeitknoten (IDE) *45*
LB_GETTEXT Nachricht *317*
LB_SETTEXT Nachricht *317*
Leere Ellipse
 Rahmenbreite bestimmen *376*
Leere Rahmen zeichnen *365*
 abgerundetes Rechteck *365*
 Ellipse *365*
 Rahmenbreite festlegen *365*
 Rahmenfarben wählen *365*
 Rechteck *365*
Leere Rahmen zeichnen (Werkzeug) *365*
 benutzen *365*
Leerer abgerundeter Rahmen

Rahmenbreite bestimmen *376*
Leerer Rahmen
Rahmenbreite bestimmen *376*
LIBRARY-Anweisung *216*
Linie
benutzen *365*
Farben wählen *365*
Strichstärke bestimmen *376*
Linie (Werkzeug) *365*
Linien Werkzeug *392*
Link-Methoden
Dynamic *48*
Static *48*
Linke Seite (Ausrichtungsoption für Dialogelemente) *305*
Linken
Kommandozeilencompiler und das *202*
Nicht-C-Module und *208*
Optionen *204*
Linken von Ressourcen *223*
Linking
von Windows-Anwendungen *48*
ListClasses Funktion in DLL Datei *293*
Listendatei *232*
Listendatei (TLIB-Option) *232*
Listenfeld *311*
Eigenschaften *311*
Load On Call (Option) *276*
LOADBWCC.EXE *490*
Locate Function, Menübefehl *153*
Logische Farbpalette *373*
Farbauflösung in Bits pro Pixel *384*
Größe *384*
reservierten Farben *384*
Logische Fehler
Definition *149*
Logische und Systemfarbpalette *384*
lokal (virtuelle Tabelle) *102*
Lokales Menü *31, 353*
Löschen
Haltepunkte *163*
Überwachte Ausdrücke *158*
Löschen der Zeichnung *363*
Löschen einer Ressource *275*
Löschen von Bezeichnern *281*
Löschen von markiertem Bereich *362*
Löschen von Symbolen *235*

M

Make *112*
(Tabelle der Direktiven *251*
Abschalten der Fehlerprüfung *253*
Ändern des benutzten Verzeichnisses *240*
Anweisungen für *242*
Aufbau aller Zieldateien *240*
Aus dem Speicher auslagern *241*
Bedingungsoperatoren *254*
Befehle *246*
Befehle für *244*
Befehls-Präfixe *246, 247*
Befehls-Präfixe (Makroerweiterung &) *247*
Befehls-Präfixe (Prüfen von Exit-Codes) *247*
Befehlsoperatoren *247*
Befehlssyntax *246*
Beschreibung von *237*
Beschreibung von BUILTINS.MAK *238*
BUILTINS.MAK *239, 240*
Dateien *240*
Definieren von Makros für *240*
Definition von *237ff*
Direktiven *251ff58*
Direktiven (!include) *254*
Direktiven (!message) *255*
Direktiven (path.ext) *255*
Direktiven(!undef) *256*
Direktiven(.precious) *255*
Direktiven(.suffixes) *256*
Erweiterter Text und *248*
Explizite Regeln *243, 244, 245, 246, 247*
Explizite Regeln (Syntax) *243*
Explizite Regeln ohne Befehle *246*
Fehler *407*
Fehlersuche *248*
Format der Regeln *243*
Hilfe per Kommandozeilenoption *238*
Hilfe-Option *238*
implizite Regeln *243, 244, 245, 246, 247*
Implizite Regeln (Verwenden mit expliziten Regeln) *246*
Kommandozeilen-Operatoren (Temporäre Datei erzeugen mit &&) *248*
Kompatibilität zu NMAKE *240*
Komponenten(:: mehrfache explizite Regeln) *244*
Laufende Bildschirmanzeige unterdrücken *240*
Leere Makros *257*
Make-Datei und *237*
Make-Dateien *242*

Make-Dateien mit anderen Namen *240*
Make-Dateien verwenden mit *239*
Makrodefinitionen löschen *240*
Makros *248, 249, 250, 251, 257*
Makros (Beschreibung der vordefinierten) *250*
Makros (Dateiname) *250*
Makros (Definieren von) *249*
Makros (Definition in Kommandozeile/in Make-Datei) *249*
Makros (Erweiterung von) *248*
Makros (Syntax für) *249*
Makros (Tabelle der Modifizierer *251*
Makros (Vordefinierte ändern) *251*
Mehrfache explizite Regeln *244*
MNehrere Ziele *242*
NMAKE-Kompatibilität *241*
Option zum Aktivieren der automatischen Abhängigkeitsprüfung *240*
Option zum Deaktivieren der automatischen Abhängigkeitsprüfung *240*
Option zum Ignorieren des Programm-Exit-Status *240*
Optionen *241*
Optionen zur Voreinstellung machen *241*
Programm-Exit-Status und *240*
Programmaufbau erzwingen *239*
Regeln *240*
Regeln für Bedingungsdirektiven *253*
Share (DOS) und *241*
Symbolische Zielangaben *242, 243*
Syntax für *237*
Syntax für implizite Regeln *245*
Temporäre Dateien *240*
Temporäre Dateien(Fehlersuche mit Hilfe von) *248*
TOUCH.EXE und *239*
Umgebungsvariablen und *240*
Unterbrechen *238*
Ünterdrücken der laufenden Bildschirmanzeige unterdrücken *240*
Verkettete Abhängigkeiten (Definition) *238*
Vordefinierte Regeln *239*
Voreingestellte Aktionen *238*
Voreinstellung der Optionen ändern *240*
Zieldateien aufbauen *242*
Zieldateien und *242*
Make-Datei
Spezifizieren der *239*
Make-Dateien
Befehle in *244, 246*
Benutzen von *242*
Fehlersuche in *248*
Fortsetzungszeilen in *244*
Implizite Regeln und *245*
MAKE-Direktiven
Bedingungs- *253*
MAKE-Optionen
-N (NMAKE-Kompatibilität) *241*
Tabelle der *240*
Verwenden von *239*
MAKEINTRESOURCE Makro in C/C++) *277*
MakeIntResource Typ in Pascal *277*
MAKER.EXE
Arbeitsspeicher und *241*
Definition von *237*
MAKESWAP.EXE (Hilfsprogramm) *2*
Makros
MAKE und *248, 249, 250*
Map-Datei/Segmente *208*
Map-Dateien
Debuggen mit *208*
Von TLINK erzeugte *208*
Markieren
Ressourcen im Projektfenster *270*
Markierter Bereich
ausrichten *377*
skalieren *377*
verschieben *377*
Markierungsfeld *295, 309, 488*
Markierungsfeld Optionen *309*
Markierungsrahmen setzen *361*
Markierungsrahmen skalieren *305*
Markierungsrahmen um Dialogelemente *296*
Markierungswerkzeug *296*
Mathematische Bibliotheken *203*
Mauspalette *31*
Symbolanzeige *39*
Mauszeiger *260*
aktiver Bereich *392*
bearbeiten *391*
Binärformat *390*
Durchsichtige Farbe *391*
eigenständige Datei *390*
Hintergrundfarbe *391*
Invertierte Farbe *391*
Kontaktpunkt *392*
Kontaktpunkt festlegen *392*
Laufzeitdarstellung *391*
Projektdatei hinzufügen *390*
Quellenform *390*

Standard Windows *389*
transparente Bereiche *370*
Verweis von Projektdatei *390*
Vordergrundfarbe *391*
Mauszeiger Datei *267*
Mauszeigerdatei erzeugen *390*
MDI (Multiple Document Interface)
Optionen *132*
MDI Client/View-Optionen
Grundoptionen *133*
Medium
Speichermodell *47*
Mehrfachpfadbibliotheken *49*
Mehrfachspeichern *283*
Meldungen
bearbeiten *351*
übersetzen *351*
Meldungsfenster *37*
Anzeige von Fehlern und Warnungen im *147*
Fehlerrückverfolgung und *38*
memchr *107*
memcmp *107*
memcpy *107*
memset *107*
Menü *259*
generisches *292*
Menü, lokales
Hilfsprogramme hinzufügen *61*
Metadateien *401*
Microsoft Compiler
Kompatibilität mit *104*
Microsoft-Windows-Anwendungen
Code-Segmente *209*
Mit den Standardfarben des Gerätes speichern
Option *378*
Modales Dialogfenster *291*
Modifizieren
Wert von Datenelementen *160*
Moduldefinitionsdatei *230*
Beispiel einer *228, 229*
CODE-Anweisung *213*
DATA-Anweisung *213*
DESCRIPTION-Anweisung *214*
EXETYPE-Anweisung *214*
EXPORTS-Anweisung *214*
IMPORTS-Anweisung *215*
LIBRARY-Anweisung *216*
NAME-Anweisung *217*
SEGMENTS-Anweisung *217*
STACKSIZE-Anweisung *218*
Standard *219*
STUB-Anweisung *218*
Moduldefinitionsdateien
\Tw, TLINK-Option und *211*
Beschreibung *212, 213, 214, 215, 216, 217, 218, 219, 220*
Groß-/Kleinschreibung und *207*
NAME-Anweisung *217*
TLINK und *212, 213, 214, 215, 216, 217, 218, 219, 220*
Module (Abschnitt in WINSPCTR.LOG) *193*
Modulnamen *234*
Moveable (Option) *276*
MS Compiler
Kompatibilität mit *104*
Muster bestimmen Dialogfenster *366*

N

Nächste Anweisung (Menübefehl) *152*
Name eines Bezeichners *277*
NAME-Anweisung *217*
Neue Berichte anhängen (Option) *188*
Neue Bitmap Attribute Dialogfenster *369*
Neue Bitmap-Attribute Dialogfenster *359*
Neue Ressource Dialogfenster *272, 403*
Neue Ressourcendatei Dialogfenster *380, 390*
Neuen Bezeichner erzeugen *402*
Neuer Bezeichner Dialogfenster *402*
Neuer Eintrag
String-Editor *357*
Neuer ResRessourcen-Typ Dialogfenster *402*
Neuer Typ Taste (Ressource Dialogfenster) *402*
Neuer-Bezeichner-Dialogfenster *280*
Neues Benutzerdefiniertes Element
Dialogfenstert *319*
Neues Bild (Befehl) *383*
Neues Symbol
Dialogfenster *380*
Neues Ziel
Dialogfenster *46*
Nicht genügend Speicherplatz (DOS-Meldung) *2*
nicht verwendet (Anzeige im Verwendung Fenster) *282*
Nicht verwendete Typen anzeigen (Option) *270*
NMAKE (Microsoft)
MAKE verwenden anstelle von *240*
No Math Support *49*

Nonportable pointer comparision 110
nonportable pointer conversion 110
NOT WS_VISIBLE Attribut 301
NOT WS_VISIBLE Nachricht 291
nspect Source (Befehl) 164
Null-Strichstärke 376

O

OBJ-Dateien 227
ObjectWindows Anwendungen:AppExpert 117
Objekte
 durchsehen in der Windows-IDE 39
Objekthierarchie
 anzeigen 39
Obsolete (Warnungen) 110
OCF 48
OEM 318
Öffnen
 Projektfenster 50
Öffnen einer Ressourcendatei 271
Öffnen eines Projekt 268
OLE 2 48
 Optionen 127
Online-Hilfe 20
OpenHelp (Dienstprogramm) 21
Operationen (TLIB-Option) 232
Optimierung 104
 ausschalten von 104
 globale 104
 lokale 104
 spezifische 104
Optimierung von Schleifen 106, 108
Optimierung von Sprüngen 105
Optimierungen
 Größen- 105
 Optionen 81
 vorkompilierte Header-Dateien 496
Optimimierung
 Geschwindigkeits- 106, 107, 108
Option
 Windows DLL 83
Optionen
 .ASM-Datei 83
 .OBJ-Datei 83
 80286 Prozessor 79
 80287 Koprozessor 80
 80386 Prozessor (16-Bit) 79
 80386 Prozessor (32-Bit) 79
 80486 Prozessor (16-Bit) 79
 80486 Prozessor (32 Bit) 79
 ANSI-Schlüsselwörter 79
 Anwendungsoptionen 125
 Bezeichnerlänge 81
 C++-Kompatibilität 81
 C-Aufrufkonvention (16-Bit) 82
 C-Aufrufkonvention (32-Bit) 83
 COMDEF 80
 Compiler 75, 79
 Compilieren ohne Linken 79
 Definieren 79
 DS != SS 81
 Elementzeiger 84
 Entry/Exit-Code 83
 Exceptions- 103
 FAR-Daten 80
 fastcall-Aufrufkonvention (16-Bit) 83
 fastcall-Aufrufkonvention (32-Bit) 83
 FastThis (16-Bit) 83
 für Projekte 65
 für Projekte ansehen 71
 Geschwindigkei 82
 Gleitkomma 80
 globale Templates 81
 Header 80
 IDE-Editor 35
 Induktionsvariablen 82
 Inline Intrinsic-Funktion 82
 Kernighan/Ritchie-Standard 79
 Laufzeit-Typinformation 83
 MDI Client/View-Optionen 133
 Optimierungen 81
 Out-of-line Inline-Funktionen 84
 Pascal-Aufrufkonvention (32-Bit) 82
 Pentium Prozessor (32-Bit) 79
 Priorotätsregeln 78
 Prozessor 79
 Register-Variablen 83
 RTTI 83
 RTTI- 103
 schnelle huge-Zeiger 81
 Speichermodell 80
 Speichermodell (virtuelle Tabellen als FAR) 84
 Speichermodell Huge 81
 Speichermodell Large 81
 Speichermodell Medium 81
 Speichermodell Medium und Never 81
 Speichermodell Small 81
 Speichermodell Small und Never 81

Speichermodell Tiny *81*
Speichermodell Tiny und Never *81*
Stack *81*
stdcall-Aufrufkonvention *83*
Tabelle *79*
Tabelle der Compiler- *79*
Template *81*
Templates- *103*
TLINK *204*
Tools *82*
überschreibende lokale *64*
UNIX V-Schlüsselwörter *79*
Verschachtelte Kommentare *79*
virtuelle Tabellen (extern) *84*
virtuelle Tabellen (lokal) *84*
virtuelle Tabellen (public) *84*
virtuelle Tabellen (smart) *84*
Windows intell. Callbacks (16-Bit) *83*
Windows Prolog-Epilog *82*
Windows-Prolog/Epilog *82*
Optionen der Botschaftsverfolgung
Dialogfenster (WinSight) *176*
Optionen/Projekt *75*
Optionen/Projekt Dialogfenster *78*
Optionen/Projekt/Linker *113*
Optionen/Speichern *35, 76*
Ordinalwert
einer Funktion *229*
Overlays
Linker und *209*
OWL *48*

P

Paint Editor
Aufruf *380*
Maustastenbelegung *370*
Paint-Editor
Arbeitsfläche teilen *367*
Arbeitsflächen-Ausschnitt *367*
Arbeitsflächen-Ausschnitt vergrößern *367*
Aufruf *359, 391*
geteilte Arbeitsfläche aktualisieren *378*
Größe der Bitmap-Ressource *359*
Statuszeile *368*
Voreinstellungen *378*
Paint-Editor aufrufen *316*
Paint-Editor,zwei Fenster benutzen *367*
Palettenindex
ansehen *368*
Pascal Aufrufkonventionen *100*
Pascalelementfunktionen *101*
Pause Program (Befehl) *154*
PDL (Project Description Language) *68*
Pentium-Anweisungen
32-Bit *99*
Pinsel *363*
benutzen *363*
Farben wählen *363*
Füllmuster bestimmen *375*
Muster wählen *363*
Pinselform bestimmen *375*
Pixel *360*
Hintergrundfarbe *360*
Vordergrundfarbe *360*
Pixelkoordinaten im Paint-Editor *392*
Plattenkapazität
ungenügende *495*
pointer comparision
nonportable *110*
pointer conversion
nonportable *110*
suspicious *110*
Portabilität *110*
Portability Warnings *109*
Position im Dialogfenster
Koordinaten *308*
Positionsanzeige *368*
Post-Mortem-Dump (Option) *189*
Präfixe für Bezeichner *279*
Pragma
hdrfile *495*
hdrstop *495*
Pragma hdrfile *495*
#pragma hdrfile *495*
Pragma hdrstop *497*
#pragma hdrstop *497*
Präprozessor Kommandos *279*
Prioritätsregeln
Optionen *78*
Programm beenden Program (Befehl) *155*
Projekt
Ansicht *55*
Befehl *64*
Beispiel *58*
Benutzerdefinierte Ressource hinzufügen *402*
benutzerdefinierte Ressourcenn einbinden *273*
Bezeichner einfügen *280*
Binärdatei *268*

compilieren *72, 268*
Dateiverweis auf Ressourcen *273*
decompilieren *268*
Einstellungen vererben *67*
Mehrfachspeichern *283*
öffnen *50, 268*
ohne Bezeichnerdatei *280*
Optionen *63*
Optionen ansehen *71*
Ressourcen direkt einbinden *272*
Ressourcendatei *268*
speichern mit Backupdateien *284*
teilweise compilieren *73*
Projekt-Manager
Dateien hinzufügen *50*
Zeichenkörper *45*
Projekt-Manager (IDE) *43*
Anlegen von Projekten *46*
Zieldateiumgebung *46*
Projekt/Compiliere *113*
Projekt/Compilieren *75, 76*
Projekt/Neues Projekt
Befehl *46*
Projekt/Projekt aktualisieren *75*
Projekt/Projekt neu compilieren *75*
Projektbaum *45*
Löschen von Quellknoten im *51*
Bearbeiten von Knoten *51*
Hierarchiediagramm *44*
Hierarchieebene *44*
Knoten hinzufügen *50*
Projektdatei *267*
erzeugen *267*
Symbol hinzufügen *380*
Typ wählen *267*
Zieldatei *46*
Projektdateien
laden *55*
umwandeln *55*
Viewing *44*
Zieldateien *44*
Ziele *64*
Projekte
anlegen *46*
anlegen *46*
Syntaxhervorhebung für *36*
Zieldateien *46*
Projektfenster
anpassen *55*
Ansicht als Datei *270*
Ansicht als Typ *270*
Ansicht Menü *269*
Bezeichner anzeigen *269*
Dateiliste *269*
Einstellung der Vorgaben *283*
Einträge anzeigen *270*
Nicht verwendete Typen anzeigen *270*
Öffnen *50*
Ressourcen auflisten *268*
Ressourcen aufzeigen *270*
Ressourcen markieren *270*
Ressourcentypen *269*
Projektknoten
Eigenschaften *51*
Löschen *51*
Option Projektansicht *44*
Projektknoten (IDE) *44*
Projektknoteneigenschaften
bearbeiten *51*
Projektoptionen, Dialogfeld *64*
Prolog/Epilog *106*
Protokoll-Ausdrücke *166*
Prozessor
32-Bit Compiler für *99*
public (Templates) *103*
public (virtuelle Tabellen) *102*
Pure (Option) *276*

Q

Quelldateien
abhängige Dateien von *44*
Laden in ein Editierfenster *45*
Viewing *45*
Zugriff auf *45*
Quelle *273, 380*
Quellenform (Microsoft kompatibel) *390*
Quellknoten
Löschen *51*
Quellknoten (IDE) *45*
bearbeiten *51*
hinzufügen *50*
Quelltext bearbeiten (Befehl) *164*
Quelltext einer Klasse
wechseln zu *142*
Quelltextdateien
separat compilierte *231*
Quelltextgrundlagen *117*
Quelltextsammlung *57*

Beispiel *58*
erstellen *58*

R

Radierer *363*
benutzen *363*
Farben wählen *363*
Maustastenbelegung *363*
Radierwerkzeug *360*
Rahmen Option (Dialogelemente Attribute) *301*
Rahmen um ein Dialogelement *301*
Rahmen-Rahmenart eines Dialogfenster *290*
Rahmenart, Dialogfenster-Attribute *289*
Rahmenbreite bestimmen *376*
RC-Dateien *222*
RC_INVOKED *222*
RCDATA Ressourcen-Typ *406*
Rechte Seite (Ausrichtungsoption für Dialogelemente) *305*
Rechteck *365*
Referenzdatei *267*, *269*
Referenzen
externe *102*
Referenzierte Ressourcen *269*, *273*
Register (Abschnitt in Protokolldatei) *192*
Register (Fenster) *162*
Registerreservierung
globale *106*
Registerwerte *162*
Reihenfolge bestimmen Werkzeug *304*
Reihenfolgenkonvention in Gruppen *303*
Reihenfolgennummern zuweisen *304*
Relativ ausrichten von Dialogelementen an Gitter *300*
rep movsw *104*
RES (Vorgabe) *283*
RES-Dateien *222*
Rescan *144*
Datenbankdateien wiederanlegen *146*
Element umbenennen *145*
Klasse importieren *145*
Klasse löschen *144*
Klasse verschieben *145*
Reset Program (Menübefehl) *155*
Resource Workshop *483*, *489*
starten aus der IDE *143*
Verwendung *1*
Verwendung unter ClassExpert *143*
Resource Workshop als Betrachter einer Anwendung *143*
Response-Datei *232*
Beispiel *236*
Response-Datei (TLIB-Option) *232*
Response-Dateien *232*
für TLINK bestimmte *201*
MAKE und *248*
TLINK und *201*
Responsedateien *77*
Ressource *259*, *260*, *261*
benutzerdefiniert *401*
Daten hinzufügen *404*
Farbenanzahl *369*
Ressourcen-Typen *401*
Ressource einfügen Dialogfenster *275*
Ressourcen *221*
Benutzerdefinierte *261*
Compilieren *222*
einbinden *262*
eingebundene *269*
erzeugen *272*
ins Projekt direkt einbinden *272*
kopieren *274*
Linken von *223*
mit Dateiverweis einfügen *274*
öffnen *271*
Quelle *273*
referenziert *273*
referenzierte *269*
Speicheroptionen *276*
Typen *259*
über Zwischenablage kopieren *275*
umbenennen *275*
Verhältnis zum Programm *259*
verschieben *274*
Ressourcen anzeigen (Option) *270*
Ressourcen auflisten *268*
Ressourcen definieren *259*
Ressourcen Editor
Rückgängig (Befehl) *283*
starten im Bezeichner-Dialogfenster *282*
Tastenbelegung *283*
Widerrufen (Befehl) *283*
Ressourcen ID *276*
Ressourcen löschen *275*
Ressourcen und Ressourcen Editor aufrufen *282*
Ressourcen-Editoren
String-Editor *352*
Ressourcen-Script-Dateien (RC) *222*

Ressourcen-Typ
benennen *402*
erzeugen *402*
ID-Wert *402*
RCDATA *406*
Ressourcen-Typen selbst definieren *401*
Ressourcen:Bezeichner *276*
Ressourcen:Binärformat *273*
Ressourcendatei *268*
Binärformat *267*
Compilieren *268*
Script-Format *267*
suchen *273*
Verzeichnis wechseln *273*
Ressourcendaten
hexadezimale Darstellung *405*
ins Ressourcenskript eingeben *405*
Projektdateien hinzufügen *404*
Richtlinien für Eingabe *405*
Ressourcenn Editor *271*
Ressourcennskript
Kommentare *272*
Ressourcenskript
Format *355*
String-Tabelle *355*
Ressourcenskript (RC) *143*
RESTYPE (Beispiel Ressourcen-Typ) *403*
RGB-Einstellung ansehen *368*
RLINK *223*
Aufruf *223*
RLINK, Optionen *223*
RLINK.EXE
Definition *199*
RLK-Kommandodatei *224*
rotl *107*
rotr *107*
Rückgängig (Befehl) *283*
Rückgängig Kommando *308*
Rückwärtskompatibilität *55, 115*
RWS_Icon-Abschnitt verändern *364*

S

Schablonen *64*
Beispiel *64*
Dialogfenster *65*
Einstellungen überschreiben *63*
mehrfache Benutzung *67*
und Knoten *67*
verwalten *66*
vordefinierte *65*
Schaltelemente *309*
3 Zustands Schaltfeld *309*
Aktionsschalter *309*
Auswahlfeld *309*
Automatisches 3-Zustands-Schaltfeld *309*
Automatisches Markierungsfeld *309*
Automatisches Schaltfeld *309*
Benutzerdefiniert *310*
Markierungsfeld *309*
Schaltfeld *309*
Selbstzeichnend *310*
Textausrichtung *310*
vordefinierte ID-Werte für Aktionsschalter unter Windows *310*
Voreingestelltes Schalter *309*
Schalter
Symbolanzeige *38*
Schaltereigenschaften Dialogfenster *309*
Schaltfeld *488*
Schere *362*
benutzen *362*
Schleife
invarianter Code in *107*
Schleifenoptimierung *106*
Schreibweise
hexadezimal *405*
hexstring *405*
oktal *405*
Schrift Datei *267*
Schrift-Befehl *365*
Schriftarten
in diesem Handbuch *19*
Schriften *261*
Schrittweiser Programmablauf *151*
Schwarzer Rahmen
Statischer Elementtyp *316*
Schwarzer Rahmen als Dialogelement *296*
SDI (Single Document Interface)
Optionen *130*
Segmente
Codesegmente komprimieren *209*
Codesegmente packen *209*
Liste der *209*
Segmentliste und ACBP-Feld *209*
TLINK und *209*
uninitialisierte bei TLINK *208*
SEGMENTS-Anweisung *217*

Seiten
Ausrichten von *206*
Seitengröße *232*
Selbstzeichnende Schaltelemente *310*
Semantikfehler
Definition *149*
Setzen von Haltepunkten *162*
Share (DOS)
MAKE und *241*
ShowWindow Funktion *301*
Sichtbar (Dialogelemente Attribut) *301*
Skalieren eines Bereichs *377*
Small
Speichermodell *47*
Soft-Modus *169, 170*
Software-Unterstützung *21*
Speed Menü *114*
Speichermodell
Compiler *81*
Speicheroptionen für Ressourcen *276*
Speicherplatzverwaltung (Option) *276*
Speicherverwaltung
mit MAKESWAP.EXE *2*
Spezielles/C++ to Assembler *114*
Sprungoptimierung *105*
st-call-Konventionen *100*
Stack switched (Meldung) *191*
Stack-Frame *81*
Stack-Frame-Daten (Option) *189*
Stack-Trace-Information in WINSPCTR.LOG *191*
Stack-Zeiger *191*
STACKSIZE *219*
STACKSIZE-Anweisung *218*
Standard Präfixe *279*
Standard-Optionen
für Projekte *65*
Standardaktionsschalter *487*
Standardbearbeitungs-Code *140*
Standardbibliotheken (IDE)
Optionen *48*
Standardtaste
Farbpalette speichern *373*
Start!
Befehl (WinSight) *172*
Starten von TLINK (Linker) *199, 200, 201, 202, 203*
Statische Dialogelemente *314*
statischer Text *489*
Statuszeile *368*
Positionsangabe *368*
Stichwortsuche *21*
Stift *363*
benutzen *363*
Farben wählen *363*
Strichstärke bestimmen *376*
Stop!
Befehl (WinSight) *172*
stpcpy *107*
strcat *107*
strchr *107*
strcmp *107*
strcpy *107*
Strichstärke bestimmen *376*
Strichstärke Null *376*
String *351*
in die String-Tabelle einfügen *354*
String IDs *354*
String-Editor
Aufruf *352*
Felder *353*
String-Tabelle *351*
bearbeiten *352*
Beispielprojekt *356*
benennen *358*
Bewegung in *355*
hexadezimale Werte *355*
Namenskonvention *358*
Ressourcenskript *355*
Speicherplatznutzung *353*
String-Editor *351*
verändern *355*
Verwaltung unter Windows *353*
Werte *353*
String-Tabellen *260*
strlen *107*
strncat *107*
strncmp *107*
strncpy *107*
strnset *107*
strrchr *107*
STUB-Anweisung *218*
STYLESHT.IDE, Beispielprojekt *64*
Suchbereich
von OpenHelp *21*
Suchen einer Funktion *153*
suspicious pointer conversion *110*
SYM-Dateien
WinSpector *191*

Symbol
3-D Effekt *386*
als Binärdatei *380*
ändern *386*
bearbeiten *381*, *386*
Bild *382*
Bild zu Ressource zufügen *382*
Binärdatei erzeugen *380*
Darstellung auf den Bildschirm *381*
Datei automatisch erzeugen *381*
eigenständige Datei erstellen *380*
erzeugen *379*
Farbanzahl bestimmen *380*
Farbvarianten in einer Ressource speichern *382*
Größe *379*
Größe bestimmen *380*
in Projektdatei einfügen *380*
invertierte Farbbereiche *382*
löschen *386*
Schattierung *385*
speichern *382*
Symboldatei erstellen *380*
transparente Farbbereiche *382*
zeichnen *384*
Symbol Datei *267*
Symbol-Attribute Dialogfenster *383*
Symbolanzeige
Filter *40*
Klassendefinition und *38*
Mauspalette *39*
Objekte *40*
Symbolanzeige (Windows-IDE)
aktivieren *39*
Symbolanzeigefenster
Arbeiten mit *38*
Symboldatei *379*, *380*
Symbole *260*
anzeigen *41*
Deklaration anzeigen *40*
Symbolfeld in Dialogfenstern *291*
Symboltabelle laden (Befehl) *168*
Symboltabellen *149*
DLLs und *168*
Wechseln der *168*
Syntax
IMPLIB *228*
TLIB *231*
TLINK *199*
Syntaxerweiterungen
hinzufügen zu *36*
Syntaxfehler *74*
ansehen *74*
Definition *147*
Syntaxhervorhebung
Dateien hinzufügen zur Anzeige mit *36*
deaktivieren *36*
modifizieren *37*
verwenden *36*
Systeminformationen in Protokolldatei *194*
Systemmenü für Dialogfenster hinzufügen *290*
Systempalette
reservierte Farben *384*
Systemtaste
Farbpalette speichern *373*

T

Tab-Taste
Bewegen im Dialogfenster *302*
Tabelle
automatische *102*
externe, virtuelle *102*
lokale, virtuelle *102*
Optionen *79*
public, virtuelle *102*
Referenzen auf *102*
Tabelle Kommando *307*
Tabellen
lokal *102*
Tabellen-Layout, bei Dialogelemente
Anordnung *306*
Tabulator Kommando *302*
Tabulator Werkzeug *302*
Tabulatorpositionen im Eigenschaften
Dialogfenster verändern *303*
Tabulatorwerkzeug *302*
TargetExpert *46*
Tasks, Abschnitt in WINSPCTR.LOG *192*
TASM *76*
Tastaturfokus *485*
Tastenbelegung im internen Texteditor *272*
Tastenkürzel *259*
konfigurieren *36*
Template
externes *103*
globales *103*
public *103*
Referenz auf *103*
Templates *103*

Test-Werkzeug *320*
Text
Farbe für *36*
Format *405*
mit Null beenden *405*
Syntaxhervorhebung (Farbgebung) *36*
Text (Werkzeug) *365*
benutzen *373, 374*
Text hervorheben *36*
Text in Ressourcen *405*
Text in Zeichnung
ausrichten *365, 374*
einfügen *365*
Farbe *374*
Farbe wählen *365*
Größe *374*
Größe festlegen *365*
Schrift bestimmen *365*
Schriftart *374*
Schrifttyp auswählen *374*
Text Werkzeug
benutzen *365*
Textbildschirm-Modus
32-Bit-Anwendung *114*
32-Bit-DLL *115*
Texteditor
Aufruf *271*
Fehler beheben mit *272*
Fehlerkorrektur mit *283*
interner *271*
Tasten *272*
Themen
aus Dialogfenstern auswählen *32*
Tiny
Speichermodell *47*
Tiny-Modell
COM-Dateien und *210*
Titel
Hinzufügen zu Dialogelementen *301*
Titel einem Dialogfenster hinzufügen *290*
Titel, Option für Dialogelemente Eigenschaften *301*
Titel-Rahmenart eines Dialogfenster *290*
TLIB *227, 231ff*
Aktionssymbole *235*
Beispiele für *235*
erweitertes Verzeichnis *233*
Fehler *407*
Kommandozeile *231*
Operationen *235*
Optionen *232*
Response-Dateien *232*
Syntax *231*
Verwendung *231*
TLINK *76, 199*
Fehler *407*
Mathematische Bibliotheken und *203*
Overlays und *209*
RLINK und *199*
TLINK (Linker)
Starten von *199, 200, 201, 202, 203*
Warnungen (Freigabe/Unterdrückung) *211*
ACBP-Feld und *209*
Bibliotheken *203*
Debug-Informationen *211*
Fehler *200*
Kommandozeilencompiler und *202*
Konfigurationsdatei *199, 201*
Konfigurationsdatei (Beispiel) *201*
Map-Datei für das ausführbare Programm, erzeugt von *208*
Nicht-C-Module und *208*
Optionen *200, 204ff*
Optionen (/A Seitenausrichtung) *206*
Optionen (/a Typ des Anwendungsprogramms) *206*
Optionen (/C Groß-/Kleinschreibung beachten) *207*
Optionen (/e Erweiterte Wörterbücher ignorieren) *207*
Optionen (/E Erweiterte Wörterbücher) *207*
Optionen (/f Erweiterte Wörterbücher) *207*
Optionen (/Gx Linker-Extras) *208*
Optionen (/i Nichtinitialisierte Endsegmente) *208*
Optionen (/l Quellprogramm-Zeilennummern) *208*
Optionen (/m Map-Dateien) *208, 209*
Optionen (/n Bibliotheken ignorieren) *208*
Optionen (/n Standardbibliotheken ignorieren) *208*
Optionen (/o Overlays) *209*
Optionen (/P Code-Segmente komprimieren) *209*
Optionen (/S Stack-Größe) *210*
Optionen (/t COM-Zieldatei) *210*
Optionen (/Td Zieloption) *211*
Optionen (/Tp Zieldatei) *211*
Optionen (/Tw Ausführbare Dateien) *211*
Optionen (/Tw Windows-Programm) *211*

Optionen (/Tw Zieldateien) *211*
Optionen (/v Debug-Information) *211*
Optionen (/w Freigabe/Unterdrückung von Warnungen *211*
Optionen (/w Freigabe/Unterdrückung von Warnungen) *211*
Optionen (/x) *212*
Optionen (/ye Expanded Memory) *212*
Optionen (/yx Extended Memory) *212*
Optionen (Zieldateien) *211*
Optionen /P Code-Segmente packen) *209*
Response-Dateien *201*
Syntax *199*
Zieldateioptionen (/Tw) *211*
TLINK Linker)
Optionen (/ye Expanded Memory) *212*
TLINK-Option (/yx Extended Memory) *212*
TLINK-Optionen
Tabelle der *204*
TLINK-Optionen (Zieldatei) *211*
TLINK.CFG *201*
Beispiel für *201*
TLINK32 *199*
RLINK und *199*
TLINK32 (Linker)
Bibliotheken *203*
Konfigurationsdatei *201*
Optionen (/B Basisadresse) *207*
Optionen (/E Maximal erlaubte Fehlerzahl) *207*
TLINK32.CFG *199, 201*
TMAPSYM
Erzeugen von SYM-Dateien mit *197*
TOOLHELP.DLL *187, 188, 189, 190, 191, 192, 193*
Tools *58*
Optionen *82*
TOUCH.EXE
Beschreibung *239*
Transfermakros *60*
Transferpuffer *141*
Transparente Farbe *382*
Transparente Farbe bestimmen (Dialogfenster) *371*
Transparente Farben
benutzen *370*
einstellen *370*
Farbdarstellung neudefinieren *371*
Voreinstellung *370*
Turbo Debugger
WinSpector und *187*
TURBO.CFG *77*
Typen
von Ressourcen *259*
Typinformationen *103*
Typumwandlung
Bezeichnerwert *277*

U

UAE-Debugger *187, 189, 191, 193, 195, 197*
Übersetzen von Meldungen *351*
Übersetzer *58*
zur IDE hinzufügen *60*
Überwachen von Ausdrücken *155*
Überwachte Ausdrücke
Aktivieren *158*
Ändern der Attribute *158*
Ändern des Anzeigeformats für *156*
Deaktivieren *158*
Hinzufügen *156*
Löschen *158*
Überwachter Ausdruck im Fenster Überwachter Ausdruck hinzufügen *156*
Umbenennen von Ressourcen *275*
Umgebung
Mehrfachpfadanwendungen *49*
Umgebungsvariablen
MAKE und *240*
Umschalt-Klicken *297*
Umwandeln
Projekt- in Make-Dateien *55*
Unbenutzte Bezeichner *281*
Untergeordnete Fenster
von Dialogfenstern nicht verdeckbar *291*
Untergeordnetes Fenster *290*
Unverändert (Ausrichtungsoption für Dialogelemente) *305*
USER- und GDI-Informationen in WINSPCTR.LOG *193*

V

Variable
Induktions- *108*
temporäre *101*
Variable untersuchen (Fenster) *155, 160*
Variablen
Modifizieren des Werts von *160*

VBX Datei *319*
Verändern
 Werte von Variablen *160*
Verbergen der Farbpalette *368, 371*
Verfolgen (WinSight)
 Botschaften *176, 177, 178, 179, 180, 181, 182*
Verfolgung (WinSight)
 Abschalten der *172*
 Steuerung der *172*
Verfolgung/Detailinformationen (WinSight) *173*
Vergrößern
 Mausaktion *362*
 Tastenkürzel *362*
Vergrößern (Werkzeug) *362*
Vergrößern Befehl *362*
Vergrößern der Zeichnung *362*
Vergrößerte Zeichnung
 bewegen in *362*
Vergrößerter Bereich
 bewegen in *366*
Verkettete Abhängigkeiten
 Definition *238*
Verkleinern der Zeichnung *362*
Verschieben einer Ressourcen *274*
Verschieben eines Bereichs *377*
Version eines Symbols erstellen *383*
Version Info *261*
Vertikal teilen (Befehl im Paint-Editor) *367*
Vertikale Trennlinie *489*
Verwendung Fenster (Bezeichner Dialogfenster) *282*
Verzeichnis
 erweitertes *233*
Verzeichnisse/Bibliothek *86*
Verzeichnisse/Exe-Dateien *86*
Verzeichnisse/Include *86*
Verzeichnisse/Obj-Dateien *86*
Verzeichnisse/Quelltext *86*
VG-Vordergrundfarbe in Farbenpalette *360*
VGA Auflösung *381*
VGA-Bildschirm *488*
View-Klasse
 TEditView *138*
 TListView *138*
 TWindowView *138*
Viewer *58*
 zur IDE hinzufügen *60*
Viewing *55*
 Projektdateien *44*
 Quelldateien *45*
Vordefinierte BWCC-Schalterelemente *487*
Vordergrundfarbe
 zeichnen *360*
Vordergrundfarbe bearbeiten (Befehl) *371*
Vordergrundfarbe benutzen *369*
Vordergrundfarbe des Mauszeigers *391*
Vorgabe
 Texteditor *283*
Vorgaben
 Ausführbar *283*
 Backup Dateien erzeugen *284*
 RES *283*
 setzen von *34*
 Windows Version *284*
Vorgaben Dialogfenster *321*
Vorgaben einstellen (Projektfenster) *283*
Vorhanden-Feld (Farbe bearbeiten) *373*
Vorkompilierte Header-Dateien
 Nachteile *494*
 Optimierung der Nutzung von *496*
 Regeln für *495*
 Steuerung *494, 495, 496, 497, 498*
Vorkompilierte Header-Dateien, Inline-Elementfunktionen und *494*

W

WantReturnFlag *487*
warn (Pragma) *109*
Warnung
 (Ineffiziente Codierungen) *112*
Warnungen *109*
 (Ineffiziente C++-Codierung) *111*
 allgemeine *112*
 Anzeige von *147*
 Auswahl von *148*
 Begrenzen der Anzahl der *148*
 definierte *408*
 im Meldungsfenster *37*
 Obsolete *110*
 Portabilitäts- *110*
 Rückverfolgung im Meldungsfenster *37*
 Standardeinstellung für *109*
Warnungen (ANSI Violations) *110*
Warnungen (Freigabe/Unterdrückung bei TLINK) *211*
Warnungen (Potentielle C++-Fehler) *111*
Warnungen (Potentielle Fehler) *111*
Watch (Debug-Menübefehl) *155*

Wdh-Zähler (Option) *157*
Werkzeugpalette *294*
- Airbrush *363*
- benutzen *360*
- Farbeimer *364*
- Gefüllte Flächen zeichnen *366*
- Hand *366*
- Leere Rahmen zeichnen *365*
- Linie *365*
- Markierungsrahmen setzen *361*
- Pinsel *363*
- Radierer *363*
- Schere *362*
- Stift *363*
- Text *365*
- Vergrößern-Werkzeug *362*
- Werkzeug auswählen *361*

Wert ändern (Bezeichner) *277*
Wert des Bezeichners ändern Dialogfenster *281*
Wert eines Bezeichners *277*
Werte
- Abfragen *155ff*

Werte wiederverwenden *108*
WHELLO, Beispiel *55*
WHELLO, Programm *52*
Widerrufen (Befehl) *283*
Wiederverwendung von *108*
WIN.INI *489*
WIN.INI-Datei *187*
Windows
- Farbauswahl *382*
- String-Tabelle ID_Werte *353*

Windows Unrevoverable Application Errors (UAEs) *187*
Windows Version (Vorgabe) *284*
Windows-Anwendungen *48*
- DS-und SS-Segmente *47*
- IDE-Optionen *48, 50*

Windows-Exit-Prozedur *228*
Windows-IDE
- editieren *35*

WinMain *490*
WinSight
- Anzeigeen in *172*
- Ausschnitte in *171*
- Auswählen von Ausschnitten in *172*
- Beschreibung von *171, 173, 175, 177, 179, 181*
- Botschaften *178*
- Klassenliste *173*
- Starten und Stoppen der Verfolgung *172*
- Starten von *171*

WINSPCTR.BIN *188*
WINSPCTR.INI-Datei *188*
WINSPCTR.LOG
- Abschnitt Register *192*
- Inhalt von *189*
- Stack-Trace-Information *191*

WinSpector
- Abschnitt Botschaftswarteschlange in WINSPCTR.LOG *192*
- Abschnitt Module in WINSPCTR.LOG *193*
- Abschnitt Tasks in WINSPCTR.LOG *192*
- Debuggen und *187*
- DFA-Dienstprogramm *195*
- DFA-Dienstprogramm (Inhalt der Datei DFA.OUT) *195*
- DFA-Dienstprogramm(Verwendung mit WINSPCTR.LOG) *195*
- Dialogfenster Vorgaben *188*
- EXEMAP *196, 197, 198*
- EXEMAP.EXE *196*
- Inhalt von WINSPCTR.LOG *189*
- Kommentare (Option) *189*
- Neue Berichte anhängen *188*
- Stack-Frame-Daten *189*
- Stack-Zeiger *191*
- SYM-Dateien *191*
- Systeminformation (Modus und Windows-Version) *194*
- TMAPSYM.EXE *197*
- USER- und GDI-Informationen *193*
- Vorige Berichte überschreiben *188*
- WIN.INI-Datei *187*
- Zusammenfassung nach AUX (Option) *189*

WinSpector (UAE-Debugger) *187, 189, 191, 193, 195, 197*
WM_COMPAREITEM *311*
WM_DELETEITEM *311*
WM_DRAWITEM *311*
WM_DRAWITEM Nachricht *310*
WM_MEASUREITEM *311*
WORKSHOP.INI-Datei *364*
WORKSHOP_INVOKED *222*
WS_VISIBLE Attribut *301*
WYSIWYG (Dialogeditor) *322*

Z

Zähler für Durchläufe *166*

Zeichenkette *351*
 mit 0 beendet *406*
Zeichnung
 bearbeiten in zwei Ansichten *367*
 Bereich ausrichten *377*
 Bereich skalieren *377*
 Bereich vergrößern *377*
 Bereich verkleinern *377*
 Bereich verschieben *377*
 Gitter auf *362*
 löschen *363*
 pixelweise bearbeiten *362*
 Text einfügen *365*
 Text hinzufügen *373, 374*
 zwei Ansichten benutzen *367*
Zeichnung vergrößern *362*
Zeichnung verkleinern *362*
Zeiger
 Alias-Namen für *104*
Zeiger auf virtuelle Tabelle ... *102*
Zeilen
 Numerierung bei TLINK *208*
Zeilennummern
 Anzeige in der Statuszeile *35*
Zentriert (Ausrichtungsoption für Dialogelemente) *305*
Zentriert im Dialog (Ausrichtungsoption für Dialogelemente) *305*
Ziehmarken *321*
Ziel
 aktualisieren *57*
Zieldatei
 Optionen festlegen *46*
Zieldateiumgebung
 Optionen *46*
Zieleigenschaften
 bearbeiten *53*
Zielknoten
 hinzufügen zu einem Projekt *52*
 löschen *53*
Zielknoten (IDE) *45*
Zieltyp
 AppExpert *52*
 Source Pool *52*
 Standard *52*
Zielumgebung
 Mehrfachpfadanwendungen *49*
Zoomfeld in Dialogfenstern *291*
Zugriff
 automatischer auf Dateien (Autodependency) *45*
 Quelldateien *45*
Zurücksetzen eines Programms *155*
Zweiter Monitor
 Benutzung mit WinSpector *189*
Zwischenablage *275*

Borland

Copyright © 1994, Borland International, Inc. Alle Rechte Vorbehalten. Alle Produktnamen von Borland sind eingetragene Warenzeichen der Borland International Inc.
Deutsche Ausgabe © 1994 Borland GmbH, Monzastraße 4c, D-63225 Langen, Telefon 06103/979-0, Fax 06103/979-290
Zweigniederlassung Schweiz, Solothurnstraße 31, CH-3303 Jegenstorf
Hauptsitz: 100 Borland Way, Scotts Valley, CA 95066-3249, +1 (408) 431-1000
Niederlassungen in: Australien, Belgien, Brasilien, Chile, Deutschland, Dänemark, Frankreich, Großbritannien, Hongkong, Italien, Japan, Kanada, Korea, Lateinamerika, Malaysia, Niederlande, Neuseeland, Schweden, Singapur, Spanien und Taiwan.
• Part # BCP1245GExxxxx